Henryk Niewodniczański Institute of Nuclear Physics,

Polish Academy of Science, Kraków

Computer simulations in high energy physics:

a case for

PHOTOS, MC-TESTER,

TAUOLA and at2sim

A dissertation for the degree of Doctor of Philosophy
by

mgr inż. **Piotr Golonka**

written under supervision of

prof. dr hab. Zbigniew Wąs

Geneva/Kraków,

21 of April, 2006

Acknowledgements:

The work presented here is built upon the experience of many people, some of them unknown to me, who put an effort in pushing computer simulations ahead in the specific area located between the phenomenology of experiments and theoretical predictions. In particular, I would like to attribute my inspiration of this thesis to ideas and projects of Alan Weinstein (CLEO), Brigite Bloch (ALEPH), Bob van Eijk (ELOISATRON), Anne Marie Lutz (ALEPH), Martin Grunewald (L3). Even though I had no opportunity to meet them, they influenced my understanding of the role of the computer simulations in High-Energy Physics, and my opinions presented in this thesis.

I am grateful for reading my manuscript and discussions on its development and early versions to: Stanisław Jadach, Elżbieta Richter-Wąs.

The work presented in the thesis, in its part concerning PHOTOS, TAUOLA and MC-TESTER, was partially supported by the EU grant MTKD-CT-2004-510126 and the Polish State Committee for Scientific Research (KBN) grant 1 P03B 091 27 for the years 2004-2006 .

The work on the *at2sim* model of the ATLAS Dataflow System was my activity during CERN Doctoral Student stipend in the CERN EP/ATR group in years 2001-2004. I'm particularly grateful to prof. Nicolas Ellis and prof. Robert W. Dobinson from CERN, and prof. P. Malecki from Cracow for providing inspiring and encouraging working atmosphere and guidance in my research. I also acknowledge a lot of help and hours of priceless discussions with dr Krzysztof Korcyl and the coleagues from the EP/ATR section at CERN. My apprenticeship in this group was an unforgettable research and personal experience.

Ultimately, I'm endlessly thankful to my beloved wife, Sylwia, and to my Mother for their patience and being the main source of motivation in completing this dissertation.

Streszczenie

Środowisko fizyki wysokich energii oczekuje z niecierpliwością i nadzieją nadejścia 2007 roku, który przynieść ma uruchomienie najpotężniejszego z do tej pory zbudowanych akceleratorów cząstek elementarnych: LHC (Large Hadron Collider - Wielki Zderzacz Hadronów) w CERN (Europejskiej Organizacji Fizyki Wysokich Energii). Cztery detektory: ATLAS, CMS, LHCb i ALICE, umieszczone na jego obwodzie, rejestrować będą wyniki czołowych zderzeń paczek protonów zachodzących 40 milionów razy na sekundę, by w trakcie późniejszych analiz, fizycy z całego świata mogli zmierzyć się z wyzwaniami dostarczenia odpowiedzi na pytania nurtujące obecnie tę dziedzinę wiedzy.

Zarówno sam akcelerator jak i detektory, których budowa zmierza obecnie ku końcowi, są owocem 20 lat pracy tysięcy fizyków i inżynierów, skupionych w eksperymentalne kolaboracje. Z uwagi na koszty, techniczną i konceptualną kompleksowość (konieczność opracowania i użycia wielu nowych, unikalnych technologii), budowa LHC i czterech eksperymentów możliwe było jedynie w wyniku globalnego, międzynarodowego wysiłku całej społeczności fizyki wysokich energii. W nie mniejszym stopniu współpraca taka będzie konieczna w ciągu następnych 20 lat pracy LHC i detektorów: analiza danych w ilościach do tej pory niespotykanych w fizycznych eksperymentach, będzie z pewnością wyzwaniem dla współczesnych i przyszłych technologii informatycznych i telekomunikacyjnych.

Symulacje komputerowe są narzędziem często stosowanym do rozwiązywania problemów o dużym stopniu skomplikowania. Techniki modelowania komputerowego są obecnie rozpowszechnione w wielu dziedzinach nauki i techniki, szczególnie zaś w pracach inżynieryjno-projektowych (np. *CAD*, *Computer-Aided-Design*). Technika budowania pomniejszonych modeli, stosowana od dawna, zastępowana jest obecnie kreowaniem komputerowych reprezenacji projektowanych obiektów i ich testowaniem. Zastosowanie technologii komputerowego projektowania pozwala uniknąć budowania prototypów w pełnej skali, co często jest nie możliwe z uwagi na, na przykład, koszty przedsięwzięcia. W przypadku eksperymentów fizyki wysokich energii, ze względu na rozmiary, kompleksowość i unikalność budowanych urządzeń, modelowanie komputerowe jest jednym z podstawowych narzędzi inżynierów i fizyków, pracujących nad projektami i konstrukcją poszczególnych elementów akceleratorów, detektorów lub stowarzyszonej z nimi infrastruktury.

Symulacje komputerowe są jednak używane nie tylko na etapie prac inżynieryjnych; w zastosowaniach naukowych odgrywają one obecnie znacznie ważniejszą rolę: pośrednika między abstrakcyjnym, formalnym językiem opisu zjawisk dostarczanych przez modele teoretyczne, a światem eksperymentów i obserwacji[1]. Postawienie znaku równości (bądź: równoważności) między przewidywaniami teoretycznych modeli[2] a wynikami obserwacji (eksperymentów) jest głównym celem fizyki. W tym

[1] Historycznie, pierwsze symulacje komputerowe przeprowadzone były dla celów zastosowań teoretycznej fizyki jądrowej w ramach projektu Manhattan (konstrukcja bomby termojądrowej), w 1946 roku. Od tego czasu, również w zastosowaniach pokojowych, symulacje komputerowe korzystające z technik Monte Carlo są nieodłącznie związane z fizyką cząstek elementarnych.

[2] W filozofii nauki często również używany jest pośredni stan opisu (i zrozumienia) wyników

celu *wszystkie* przewidywania teoretyczne, mogące mieć wpływ na wynik obserwacji muszą być wzięte pod uwagę (i ich efekt zidentyfikowany) przy analizie wyników eksperymentu. W przypadku eksperymentów fizyki wysokich energii, wielość procesów, które trzeba wziąć pod uwagę (proces "twardy", QCD, QED, rozpady cząstek, korelacje spinowe, ...) oraz kompleksowość samych eksperymentów (efektywność detektorów, ich geometria, oddziaływanie cząstek z materiałem detektora,...), wyklucza obecnie możliwość dokonania analizy danych w sposób analityczny. Biorąc pod uwagę fakt, iż większość badanych obiektów (cząstek produkowanych w zderzeniach, opisywanych modelami teoretycznymi) nie jest bezpośrednio rejestrowana w detektorach (jedynie produkty ich rozpadu), a część z nich jest zupełnie nieobserwowalna (kwarki), jedyną możliwością dowiedzenia ich istnienia i podania właściwości jest konfrontacja obserwabli zrekonstruowanych z własności zarejestrowanych cząstek z symulacjami komputerowymi, które bazowane są na łańcuchu modeli opisujących procesy prowadzące do zarejestrowania końcowych cząstek w detektorach. Częściowe symulacje, z których następnie składa się kompletną, muszą zawierać opisy m.in. teoretycznego modelu interakcji zderzancyh cząstek, bazowanego na teoriach pola (QED, QCD) oraz modelach fenomenologicznych (hadronizacja), propagacji wyprodukowanych cząstek w materiale detektora i związanych z tym efektów (zakrzywienie torów w polu magnetycznym, konwersja, promieniowanie przejścia itp.), detekcji cząstek (określona energetyczna i geometryczna zdolność rozdzielcza detektora, elektronika odczytu). Zidentyfikowanie relacji odpowiedniości w analizach przeprowadzanych na danych eskperymentalnych i danych pochodzących z łańcucha symulacji komputerowych jest równoznaczne z dostarczeniem ostatecznego wyniku eksperymentu i interpretacji.

W przedstawianej pracy doktorskiej rozległy temat symulacji komputerowych w fizyce wysokich energii jest prezentowany od kilku stron, nie pretenduje jednak do kompletnego, wyczerpującego opisu problematyki. Dyskusje i prezentowane poglądy są owocem mojej bliko dziesięcioletniej współpracy z wieloma grupami fizyków, pracujących na różnych etapach rozwoju eksperymentów, jak i z teoretykami. Dzięki różnorodności kontaktów z grupami stosującymi techniki symulacji komputerowych, byłem w stanie stworzyć własny, dość szeroki, obraz problematyki i wypracować ogólne spojrzenie na prezentowaną tematykę. Tak jak miało to miejsce w przeszłości, poglądy i opinie, które przezentuję mogą ulec ewolucji. Nie mniej jednak wierzę, iż materiał zawarty w tej pracy może być pomocny, dla zainteresowanych czytelników, przy formowaniu własnego spojrzenia i opinii dotyczących zastosowań symulacji komputerowych w fizyce wysokich energii.

Pierwsza część doktoratu jest wielowątkowym wprowadzeniem i przedstawia ogólne spojrzenie na temat symulacji komputerowych. Zawiera ona również informacje ogólne, takie jak opis programu fizycznego i detektora ATLAS (rozdział 5). Eskperyment ATLAS wyrałem jako przykładowy w mojej pracy doktorskiej, z uwagi na zaangażowanie krakowskiego środowiska fizycznego w tę kolaborację, jak również moją blisko z tym eksperymentem związaną ścieżkę edukacji i pracy naukowej (stypendium CERN Doctoral Student w latach 2001-2003 w grupie zajmującej się systemem triggera wysokiego poziomu i zbierania danych dla eksperymentu ATLAS). W rozdziale 2 dyskutowane jest miejsce symulacji komputerowych w metodologii nauki. Zgodnie z wieloma obecnie wyrażanymi tezami, popieram pogląd w którym są one równoprawnym elementem dostępnej nam metodologii, komplementarnym z opisem teoretycznym i eksperymentem (obserwacją). Przedstawiona i uwypuklona jest rola walidacji (weryfikacji poprawności, konfrontacji z rzeczywistością) symulacji, w szczególności w kontekście zastosowań naukowych (sekcja 2.10). Kuliminacją rozdziału jest przedstawienie, w sekcji 2.7, metodologii symulacji

obserwacji, nie będący bezpośrednimi implikacjami modeli teoretycznych, wyrażany terminem "fenomenologia".

komputerowych, która jest owocem syntezy mojej własnej praktyki i doświadczeń oraz przesłanek z wielu źródeł. Specyfikacja tej metodologii, która jest potem używana konsystentnie w dalszych częściach pracy, jest pierwszym z moich własnych zaprezentowanych tu osiągnięć poznawczych. W rozdziale 3 przedstawione są konteksty, w jakich symulacje komputerowe są używane w fizyce wysokich energii, a w szczególności: konieczność stosowania skomplikowanych łańcuchów symulacyjnych jak również zagadnienia związane z fizyczną wartością kodów symulacji komputerowych. Rozdział 4 jest przedstawieniem metodologii programowania opartej o komponenty i jej relacji z fizycznymi symulacjami komputerowymi. Jest to również wprowadzenie do dyskusji nad często pomijanym problemem sposobu dekompozycji skomplikowanych modeli na modele cząstkowe, związanych z tym faktem aproksymacji i fizycznej precyzji wyników.

W części *II* prezentowany jest model `at2sim` Systemu Akwizycji Danych eksperymentu ATLAS. Opracowanie modelu, jego kalibracja i walidacja oraz (częściowo) przeprowadzenie symulacji kompletnego systemu przy użyciu modelu `at2sim` było tematem mojego stypendium *CERN Doctoral Student* (w latach 2001-2003) i jedną z głównych linii mojej aktywności naukowej w tym czasie. Weryfikacja architektury Systemu Akwizycji Danych przez symulacje komputerowe była ważnym etapem w procesie projektowania tego systemu, wymaganym przez raport TDR [70], akceptacja którego była równoznaczna z ostateczną zgodą na konstrukcję systemu w proponowanym kształcie.

System Akwizycji i Filtrowania Danych eksperymentu ATLAS oparty jest o rozległą sieć komputerową w technologii Gigabit Ethernet, łączącą kilka tysięcy elementów będących buforami danych (od strony detektora) i elementami przetwarzającymi, które z technicznego punktu widzenia są standardowymi (dwu-procesorowymi) komputerami pracującymi w systemie Linux i wykonującymi specjalizowane, wielowątkowe programy (Dataflow Software). Sieć łącząca komponenty oparta jest o swicth'e (przełączniki sieciowe) ułożone w wartswy, koncentrujące funkcjonalne elementy systemu.

Symulacje komputerowe miały za zadanie zbadanie zachowania systemu w pełnej skali na podstawie testów możliwych do przeprowadzenia w warunkach laboratoryjnych (zbudowanych z pojedynczych, dostępnych komponentów stanowiących 1-10% kompletnego systemu). W pierwszym przybliżeniu analityczne modele oparte o średnie częstości zdarzeń i przepustowości poszczególnych elementów mogły dostarczyć "statycznych" informacji o stanie systemu pracującego w warunkach stabilnych. Model statyczny (tzw. "*paper model*") nie był jednak w stanie zbadać dynamiki systemu w sytuacjach niestacjonarnych, które występują w systemach filtrowania i akwizycji danych (fluktuacje w częstości zdarzeń wynikające z ich niezależności). Z tego powodu zostały opracowane dwa dynamiczne modele: `at2sim` i `Simdaq` (przewidywania obydwu, w najważniejszych przypadkach, mogły być skonfrontowane ze sobą dla weryfikacji). Oparte są one o modelowanie w dziedzinie zdarzeń dyskretnych (*discrete event*) i użyte, by zlokalizować potencjalne "słabe punkty" w proponowanych architekturach, w których dojść mogłoby do zagubienia danych bądź słabej wydajności. W tym kontekscie, wyniki symulacji kierowały do pewnego stopnia rozwojem komponentów Systemu Akwizycji Danych i dostarczały wskazówek dla optymalizacji ich wydajności. Symulacje dostarczyć miały również informacji o średniej zajętości poszczególnych elementów systemu, implikowanej koniecznością transmisji danych. Pozostająca część mocy obliczeniowej, dla elementów będących procesorami poziomu LVL2 triggera, miała być bezpośrednio wykorzystana przez algorytmy selekcji przypadków. Z tego względu znajomość obciążenia procesorów wskutek komunikacji przekładała się na ilość elemetów systemu koniecznych, by przetworzyć na czas dane pochodzące z detektora. W ten sposób mogła zostać określona ostateczna architekturę (ilość, moc obliczeniowa komputerów) jak i para-

metry które będzie mógł osiągnąć system przetwarzania danych.

Pierwotna wersja modelu `at2sim` została opracowana zanim dołączyłem do grupy zajmującej się modelowaniem systemu akwizycji danych eksperymentu ATLAS. Głównym jego osiągnięciem było poprawne modelowanie wczesnych prototypów systemu, testowanych w laboratorium, co pozwoliło utwierdzić się w przekonaniu o prawidłowym wyborze narzędzi (pakiet `Ptolemy`, opisany w rozdziale 8) i ogólnej metodologii symulacji. Ważnym osiągnięciem było również opracowanie parametryzowalnego modelu przełącznika sieciowego (switch'a) oraz metodologii jego kalibracji.

Mimo wielu sukcesów w modelowaniu sieci i przełączników sieciowych, początkowa wersja modelu `at2sim` była jednakże zbyt uproszczona, by poprawnie modelować zachowanie elementów zbudowanych przy użyciu (dwuprocesorowych) komputerów wykonującyhc ostateczne (nie-prototypowe) wersje oprogramowania Systemu Akwizycji Danych. Nowa wersja symulacji `at2sim`, której byłem głównym autorem, miała na celu opracowanie modeli elementów, których precyzja byłaby wystarczająca do dokładnych symulacji systemów testowanych w laboratoriach, a jednocześnie na tyle prostych, by możliwe było przy ich pomocy zasymulowanie systemu w pełnej skali. Pierwotna wersja symulacji `at2sim` oraz argumenty uzasadniające konieczność stworzenia nowej wersji przedstawione są w rozdziale 6.

Opracowanie nowej wersji modelu `at2sim` wymagało zrozumienia wydajności i mechanizmów wszystkich elementów systemu akwizycji danych; jedynie model przełącznika sieciowego, doskonale symulujący niemal wszyskie zależności i charakterystyki rzeczywistego switch'a, mógł być przeniesiony do nowej wersji. W testach, jakie przeprowadziłem na układach laboratoryjnych, stwierdziłem, iż wydajność wszystkich elementów zdeterminowana jest w znacznym stopniu przez operacje związane z transmisją danych za pośrednictwem sieci, które - przy wymaganych przepustowościach transmisji - zużywają znaczące ilości zasobów procesora. Z tego względu wstępnym etapem przygotowującym nową wersję modelu `at2sim` były gruntowne studia mechanizmów odpowiedzialnych za transmisję sieciową w systemie operacyjnym Linux, połączone z testami pozwalającymi ocenić wpływ poszczególnych jego elementów na wydajność systemu. Technologia Gigabit Ethernet, która stała się głównym kandydatem medium sieciowego, przyniosła dodatkowe mechanizmy pozwalające w znaczący sposób zoptymalizować przesyłanie danych przy dużych przepustowościach (techniki mitygacji przerwań). Jednakże użycie tychże technik nie było w tamtym czasie wystarczająco udokumetowane. Z tego też względu konieczne było przeprowadzanie wielu eksperymentów z dziedziny optymalizacji bezstratnej transmisji danych pomiędzy dwoma (i więcej) komputerami. Przeprowadzenie testów wiązało się z koniecznością wypracowania mojej własnej metodologii prowadzenia testów (programy testowe, konfiguracja komputerów, systemu operacyjnego, w tym własne, optymalizowane wersje jądra systemu, konfiguracja kart sieciowych i połączeń, automatyzacja wykonania długich serii testów, detekcja błędów i utraconych danych), jak i analizy wyników. Metodologia, najważniejsze wyniki i wnioski (implikujące bezpośrednio architekturę nowego modelu) przedstawione zostały w rozdziale 7.

Zrozumienie wydajności mechanizmów transmisji danych między komputerami pozwoliło na opracowanie ogólnego modelu wielowątkowej aplikacji sieciowej pracującej na (jedno- lub wieloprocesorowym) komputerze, zaopatrzonym w jeden lub wiele interfejsów sieciowych. Model ten, stanowiący podstawę wszystkich modeli nowej wersji `at2sim`'a, oparty jest na współdzielonych zasobach czasu procesora, zużywanych przez system operacyjny oraz poszczególne, równolegle wykonujące się wątki aplikacji. Bazując na wynikach pomiarów przeprowadzonych wcześniej, możliwe było skalibrowanie parametrów odpowiedzialnych za proces transmisji i odbioru danych z sieci.

Wykorzystanie architekury opartej o komponenty i technik programowania obiek-

towego, pozowoliło następnie w stosunkowo prosty sposób stworzyć modele poszczególnych elementów systemu akwizycji danych, przez rozszerzenie modelu ogólnego o elementy funkcjonalne poszczególnych aplikacji. Dzięki zastosowaniu takiego podejścia również pomiary kalibrujące dla poszczególnych modeli zostały znacząco uproszczone: w pierwszym przybliżeniu kalibracja modelu ogólnego okazywała się dawać dobre rezultaty. Wartości, dla dodatkowych parametrów poszczególnych modeli, których wprowadzenie okazywało się konieczne, były znacznie prostsze do uzyskania niż kalibracja modelu ogólnego. Instrumentacja kodu aplikacji i przeprowadzenie analizy śladów jego wykonania programu okazywało się często wystarczające. Modele elementów (modelu ogólnego, jak i właściwych aplikacji systemu akwizycji danych) zaimplementowane w `at2sim`, proces kalibracji i walidacji tych modeli oraz przykłady wyników uzyskanych przy użyciu `at2sim` są opisane w rozdziale 9.

Moja aktywność naukowa związana z modelem `at2sim` zakończyła się w 2003 roku, na etapie, gdy modele poszczególnych elementów systemu zostały skompletowane i skalibrowane, zaś poprawność zweryfikowana w procesach walidacji (porównaniach wyników symulacji z wynikami pomiarów przeprowadzonymi na stanowiskach testowych). Model został przekazany w ręce osób odpowiedzialnych za symulacje i analizy wydajności systemu akwizycji danych w pełnej skali. Od tego czasu jest używany, by wciąż zwiększać zaufanie do zaakceptowanej architektury systemu i poszukiwać alternatywnych możliwości jego dalszych ulepszeń.

Kolejne dwie części pracy doktorskiej, *III*, *IV* nawiązują do hasła, będącego tytułem książki, jednego z pionierów informatyki, N. Wirtha: "*Algorytmy + Struktury Danych = Programy*" [35]. Mimo trzech dekad które minęły od jej wydania i kilku rewolucji zmieniających poglądy i metodologie pisania programów komputerowych, wiele z uwag wyrażonych w tej, archiwalnej dziś książce jest nadal trafnych i stosowalnych obecnie.

Tematyką trzeciej części prezentowanej rozprawy doktorskiej są struktury danych specyficzne dla łańcuchów symulacji Monte Carlo w fizyce wysokich energii - tzw. *event record*. Od czasów eksperymentów prowadzonych na akceleratorze LEP klasyczną architekturą budowania skomplikowanych łańcuchów symulacyjnych jest połączenie kodów wielu programów (bibliotek), dedykowanych do symulacji poszczególnych elementów, za pomocą wspólnej struktury danych, event record ("rekord zdarzenia"). Spełnia ona rolę interfejsu pozwalającego na przekazywanie danych między modułami. Jest również miejscem zapisu cząstkowych aktywności każdego z programów, jak i składowania ostatecznych wyników symulacji pojedynczego przypadku. Poszczególne moduły współpracują w symulowaniu reprezentacji każdego przypadku fizycznego (wyniku zderzenia cząstek), pobierając z niego dane wyprodukowane przez inne moduły i zapisując własne cząstkowe wyniki. Taki sposób modularnej organizacji symulacji wymaga zdefiniowania standardu dla struktury event record tak, by każdy z modułów mógł dostarczyć standardowych mechanizmów wymiany danych przy jego pomocy. W roku 1991 zaproponowany został standard HEPEVT dla struktur event record. Standard ten został szybko zaakceptowany przez autorów większości kodów symulacji Monte Carlo, co ugruntowało na wiele lat modularną architekturę łańcuchów symulacyjnych z centralną strukturą event record. Podstawowym problemem standardu HEPEVT okazał się jednak brak możliwości jego rozszerzeń, motywowanych wymaganiami nowych modeli fizycznych. Dla nowych zastosowań (LEP-II) standard HEPEVT został zmodyfikowany, by możliwe stało się składowanie w nim opisów większej ilości cząstek, z większą precyzją. Zaczęły się pojawiać niezgodne ze sobą wersje struktury HEPEVT. Co więcej, konieczność składowania dodatkowej informacji (takiej jak korelacje spinowe bądź kolor), które nie były zdefiniowane w oryginalnym standardzie, zmusiła autorów symulacji do stosowania własnych konwencji wypełniania danymi struktury

HEPEVT, często łamiących w nieodwracalny sposób gramatykę samej struktury i skutecznie uniemożliwiających stosowanie HEPEVT jako standardu. Mimo tych problemów, HEPEVT przetrwał jako szeroko używany standard do czasów LHC i, obarczony wspomnianymi problemami, jest dalej stosowany w większości kodów generatorów przypadków. Akceptacja języka programowania C++ przez kolaboracje eksperymentalne na LHC wydawało się być znakiem końca ery kodów symulacyjnych w języku FORTRAN, i końcem struktury HEPEVT. Potencjał rozszerzalności i abstrakcji dostępny przy zastosowaniu obiektowo zorientowanych technik programowania mógł być sposobem rozwiązania problemów, z którymi borykała się struktura HEPEVT. Jednakże trend programowania obiektowego, zaakceptowany przez nowe eksperymentalne kolaboracje, nie udzielił się autorom większości istniejących modułów symulacyjnych, zwłaszcza tych odpowiedzialnych za symulacje modeli teoretycznych i fenomenologicznych. Najważniejszym z powodów, które powstrzymały przyjęcie nowych technologii programowania była konieczność utrzymania i zachowania istniejących i wciąż używanych kodów, których wartość fizyczna, wypracowana przez długie lata walidacji (przez porównania z danymi eksperymentalnymi wielu eksperymentalnych kolaboracji) miała nieporównywalnie większą fizyczną wartość niż potencjalne korzyści które mogłyby płynąć z przepisania kodów w języku C++. Co więcej, nowe kody symulacyjne, pisane w C++ również nie mogły wykorzystać potencjału płynącego z programowania obiektowego przy wymianie danych i organizacji interfejsów; by być wykorzystanymi jako moduł w łańcuchu symulacyjnym musiały one dostarczyć interfejsów do struktury HEPEVT. Projekty przenoszenia istniejących symulacji z języka FORTRAN do C++, na tym etapie okazały się być bezcelowe - osobiście doświadczyłem tego przy projekcie `Photos+` [56], implementacji w C++ algorytmu `PHOTOS`, który, z uwagi na konieczność wymiany danych jedynie za pomocą standardu HEPEVT, nie zastąpił swej oryginalnej wersji w języku FORTRAN. Brak standardu dla struktury event record w języku C++ okazał się, jednocześnie, być spowodowanym i być przyczyną hamującą rozwój nowych kodów symulacyjnych (opartych o architekturę z centralną strukturą event record) w C++. Propozycje takich struktur w oczywisty sposób zaczęły się pojawiać, jednakże żadna z nich nie spotkała się z szeroką akceptacją środowiska autorów symulacji - żadna z nich nie stała się standardem na miarę HEPEVT. Nie rozwiązywały one w zadowalający sposób większości problemów oryginalnego standardu; większość z nich odpowiadała zapotrzebowanieom jedynie jednej z zaangażowanych grup (np. generatorów przypadków), pomijając zupełnie kwestie wymiany danych z innymi modułami. Brak grupy, bądź organizacji o wystarczającym autorytecie, mogącej zdefiniować nowy standard i wyegzekwować jego akceptację jest również jedną z przyczyn dla której standard struktury event record w C++ wciąż nie istnieje. Brak jednego, standardowego rozwiązania oznacza konieczność dostarczania wielu interfejsów do istniejących struktur - zarówno tych w języku FORTRAN jak i C++. Moim własnym rozwiązaniem, które okazało się skuteczne w przypadku programu `MC-TESTER`, jest biblioteka `HEPEventLib`, będąca abstrakcyjną warstwą prezentującą dane zawarte w event record, wraz z grupą interfejsów potrafiących wymieniać dane pomiędzy tą warstwą a implementacjami różnych standardów. Dzięki zastosowaniu `HEPEventLib`, kody symulacji mogą się ograniczyć do pojedynczego interfejsu (z abstrakcją warstwy `HEPEventLib`) i korzystać z danych wszystkich standardów, które obecnie bądź w przyszłości będą obsługiwane przez `HEPEventLib`.

Najważniejszymi osiągnięciami przedstawionymi w części *III* są biblioteka HEPEventLib oraz zgromadzenie informacji potrzebnych do dyskusji (i propozycji implementacji) nowego standardu dla struktury event record w C++. W rozdziale 10 zgrupowano informacje ogólne oraz wstępną listę wymagań, które spełnić powinien nowy standard. W rozdziale 11 zawarte jest podsumowanie najważniejszych istniejących rozwiązań, wraz z ich silnymi i słabymi stronami. Informacje zawarte w tej części doktoratu nie pretendują do tytułu pełnej specyfikacji wymagań dla

nowego standardu. Moim zamierzeniem było informacji i argumentów do dyskusji nad możliwością definicji nowego, wspólnego standardu, z grupami zajmującymi się tą tematyką.

Czwarta część pracy jest poświęcona algorytmom używanym w modułach łańcuchów symulacyjnych. W rozdziale 12 przedstawiam uproszczony opis etapów generacji pojedynczego przypadku fizycznego. Celem tego rozdziału jest dostarczenie opisu który jest dostępny pojęciowo dla czytelników zainteresowanych całościowym, intuicyjnym zrozumieniem procesu symulowania kompletnych przypadków oraz zależnościami między poszczególnymi etapami. W chwili obecnej nie są mi znane tego typu przeglądowe opisy[3] dostępne dla czytelników, którzy mogą czuć się zagubieni w zawiłościach teorii pola i rachunku perturbacyjnego - narzędzi o dużym stopniu matematycznej abstrakcji. Jednocześnie prezentowane opisy powinny być motywacją do samodzielnych poszukiwań bardziej aktualnych i kompletnych informacji. Wierzę, że usunięcie podstawowej bariery pojęciowej, jest istotnym elementem zaangażowania w dyskusję osób, nie będących ekspertami w dziedzinie.

Rozdział 13 jest prezentacją mojego zaangażowania w rozwój modułu symulacyjnego `PHOTOS`, generującego poprawki promieniste QED w rozpadach cząstek. Jest on interesującym przykładem połączenia wielu technik na poziomie modelu koncepcyjnego, jak i na poziomie implementacji, zaś jego modularna forma i umiarkowana wielkość kodu, pozwalają ogarnąć jego zawartość. `PHOTOS` jest uniwersalnym modułem Monte Carlo, stosowanym w łańcuchach symulacyjnych wielu eksperymentalnych kolaboracji jako "moduł-dopalacz" generujący dodatkowe fotony, których istnienie przewiduje elektrodynamika kwantowa. Jednakże z uwagi na konieczność uproszczenia modeli i ich stosunkowo niewielki wpływ na większość wyników (rzedu 1%), fotony te nie są brane pod uwagę w większości kodów symulacyjnych. W eksperymentach dokonujących obecnie precyzyjnych pomiarów, takich jak Belle, BaBar, wpływają jednak na wyniki eksperymentalne, i ich uwzględnienie jest konieczne przy analizie. Podejście precyzyjne wymaga zwykle napisania dedykowanego, precyzyjnego modułu symulacyjnego dla analizowanego procesu, biorącego pod uwagę, między innymi poprawki promieniste QED. Alternatywą jest zastosowanie już istniejących modułów i "dogenerowanie" fotonów przy pomocy `PHOTOS`'a. W tym wypadku rzeczywista precyzja fizyczna efektywnego, złożonego modelu okazała sie być niewystarczająca dla niektótych zastosowań (temat ten jest obszerniej dyskutowany w części VI). Zastosowania `PHOTOS`'a w reżimie pomiarów o dużej precyzji skłoniły autorów do wprowadzenia znaczących ulepszeń zarówno technicznych (numeryczna stabilność), jak i koncepcyjnych (uniwersalna waga interferencyjna, efekty pełnych elementów macierzowych). Dzięki wprowadzeniu poprawek udało się wyeliminować większość problemów typowo napotykanych przez użytkowników `PHOTOS`'a, oraz znacząco powiększyć zakres jego stosowalności. Tematyka ta wykracza jednak poza obecną pracę.

Rozdział 14 dokumentuje moją działalność związaną z generatorem `TAUOLA` oraz jego `Uniwersalnym Interfejsem`

do struktury HEPEVT. `TAUOLA` jest biblioteką generującą rozpady τ, opartą o modele uwzględniające korelacje spinowe. Teoretycznyczne i fenomenologiczne modele rozpadów leptonów τ, na których bazują algorytmy `TAUOLI` zostały wcześniej szczegółowo opisane m.in. w pracach [2,3], dlatego też zdecydowałem nie powtarzać tych opisów tutaj, mimo iż byłem zaangażowany w badania tam opisane, głównie na etapie konstrukcji narzędzi i projektowania testów. Przedstawiam natomiast tematykę aplikacji `TAUOLI` w dwu, różniących się fizyczną precyzją, interfejsach do zewnętrznych generatorów odpowiedzialnych za produkcję leptonów τ. Tematyka

[3] z wyjątkiem transparencji wykładów, które można znaleźć w Internecie; materiał w nich zawarty zwykle prezentuje tematykę dość jednostronnie - z punktu widzenia jednego autora.

wpływu specyfiki interfejsów TAUOLI na organizację architektur łańcuchów symulacyjnych nie była wcześniej opisana w sposób wystarczający.

Z faktu, iż algorytmy generatora TAUOLA oparte są o modele biorące pod uwagę spin (polaryzację) leptonów τ, wynika konieczność odmiennej organizacji interfejsów do zewnętrznych modułów symulacyjnych. Od strony fizycznej bezpośrednią przyczyną są prawa mechaniki kwantowej, które nie pozwalają, by (przy poprawnym symulowania pełnych efektów spinowych, bez aproksymacji) procesy produkcji i rozpadów cząstek traktowano oddzielnie. W procesach opisywanych przez Model Standardowy, uwzględnienie spinu jest wymagane w procesach z udziałem leptonów τ i kwarków t. W innych przypadkach aproksymacje pozwalające rozdzielić te dwa procesy, dają satysfakcjonujące wyniki lub konstruowane są programy symulujące jednocześnie proces produkcji i rozpadu. Z tego względu klasyczna architektura oparta o niezależne moduły symulacyjne połączone centralną strukturą event record, nie może być zastosowana do rozdzielenia procesu produkcji i rozpadów leptonów τ. Często taka dekompozycja ma miejsce dla innych cząstek: np. kaskadowe rozpady cząstek i rezonansów które są generowane przez inny moduł niż ten który odpowiedzialny był za generowanie "twardego procesu". Organizacja interfejsu modułu TAUOLA jest więc odmienna od typowej organizacji bazowanej na strukturze HEPEVT. Ma ona formę specyfikacji interfejsu w postaci nazw i parametrów funkcji, które muszą być dostarczone przez użytkownika, by przekazać informację na temat leptonów τ i ich produktów rozpadu. Dla generatorów takich jak KKMC, które współpracują z TAUOLĄ przekazując pełną informację na temat spinu, odpowiednie interfejsy są dostarczane przez sam generator. W pewnych przypadkach pożądane jest jedak użycie TAUOLI do generacji rozpadów tauonów, nawet jeśli efekty spinowe mogą być potraktowane jedynie w przybliżony sposób. Aby to umożliwić stworzony został TAUOLA Universal Interface, pozwalający używać TAUOLI w architekturach opartych o centralny event record, HEPEVT. Dla niektórych procesów przybliżona informacja o spinie cząstek może być wyliczona z informacji dotyczącej typu cząstek, z których powstały leptony τ. W tych przypadkach możliwe jest generowanie przybliżonych, bądź nawet pełnych, informacji spinowych. Od strony fizycznej tematyka ta jest szerzej dyskutowana np. w [2]. Od strony technicznej, organizacja wymiany danych pomiędzy TAUOLĄ a HEPEVT została opisana w powyższym rozdziale.

Piąta część jest poświęcona narzędziom, których używałem w moich pracach związanych z symulacjami komputerowymi. W rozdziale 15 podsumowuję własne doświadczenia z narzędziami i praktykami ich użycia, wspomagającymi programowanie, jak i organizację projektów programistycznych. Dostępność wielu, nieznanych szerzej, nowych narzędzi ułatwiających programowanie i lokalizację błędów, pozwalało na znaczne ułatwienie technicznych etapów rozwoju symulacji.

W kolejnych dwu rozdziałach prezentuję narzędzia własnego autorstwa, których konteksty użycia leżą na pograniczu fizyki i rozwoju projektów symulacji komputerowych.

Pierwszy z tych projektów, TAUOLA-PHOTOS-F, opisany w rozdziale 16, jest formą narzędzia służącego organizacji projektów programistycznych. Głównym jego celem jest utrzymywanie istniejących wersji generatorów TAUOLA i PHOTOS i dostarczanie kodów źródłowych tych generatorów w wersjach i opcjach wyspecyfikowanych przez zaawansowanych użytkowników tychże pakietów (typowo osób odpowiedzialnych za instalacje tych pakietów w łańcuchach symulacyjnych kolaboracji eksperymentalnych), co wymaga uzgodnienia interfejsów i fizycznych inicjalizacji w kodach. Środowisko TAUOLA-PHOTOS-F pełni rolę archiwum konserwującego istniejące warianty generatorów, z ich fizycznymi inicjalizacjami dostarczonymi przez kolaboracje eksperymentalne oraz interfejsy. Zapewnia również możliwość rozwoju i integracji nowych wersji (w przypadku TAUOLI, nowe wersje są owocem współpracy z kolaboracjami

Belle i BaBar) do tegoż środowiska.

Ewolucja narzędzi programistycznych, jak i architektur i systemów operacyjnych, była motywacją dla restrukturyzacji infrastruktury potrzebnej do kompilacji generatorów TAUOLA i PHOTOS, aby umożliwić łatwiejszą adaptację tych pakietów w przyszłości. Z praktyki autorów wynika, iż narzędzia takie jak kompilatory bądź pre-procesory wymagają wyspecyfikowania opcji różniących się w zależności od użytej platformy i wersji systemu operacyjnego. Opracowanie nowej, prostej w użyciu i rozszerzalnej architektury, użytej później również w pakiecie MC-TESTER, umożliwiło kompilację TAUOLI i PHOTOS'a na różnych platformach przez ostatnie lata. Aktywność w tym kierunku była później kontynuowana w studiach narzędzi, dyskutowanych w sekcji 15.2.

Konieczność archiwizowania i utrzymania kodów o znaczącej zawartości fizycznej (inicjalizacje i parametryzacje modeli w generatorze TAUOLA, zawierające fenomenologiczne modele i dopasowania do eksperymentalnych danych) jest motywowana przez potrzeby porównywania, weryfikowania i kombinowania wyników pomiędzy kolaboracjami eksperymentalnymi - tymi, które zakończyły już proces analizy swoich danych (np. ALEPH, CLEO) oraz tymi, które takie analizy przeprowadzają obecnie (Belle, BaBar) lub będą przeprowadzać w przyszłości (LHCb). Możliwość dostarczenia potrzebnej wersji kodu, ze źródeł wiadomego pochodzenia i zawartości fizycznej, tj. od samych autorów pakietu, jest w tym wypadku wielkim ułatwieniem dla koordynatorów oprogramowania w kolaboracjach eksperymentalnych. Z drugiej strony, dla samych autorów, pakiet TAUOLA-PHOTOS-F jest niezbędnym krokiem w stronę realizacji projektu reorganizacji kodu i jego translacji do języka C++. Utrzymanie oryginalnej implementacji, mogącej służyć jako wersja referencyjna jest niezbędne, aby translacja taka zachowała fizyczną jakość symulacji, wypracowaną przez lata walidacji w porównaniach z danymi eksperymentalnymi. Narzędzie MC-TESTER, opisane poniżej, z pewnością pełnić będzie ważną rolę w planowanych procesach translacji.

MC-TESTER, opisany w rodziale 17, jest półautomatycznym narzędziem pozwalającym przeprowadzać testy zgodności wyników produkowanych przez programy symulujące procesy rozpadów cząstek. Początkowo miał on być elementem środowiska TAUOLA-PHOTOS-F, pozwalającym przeprowadzać testy walidacyjne wariantów generatora TAUOLA dostarczanych przez kolaboracje eksperymentalne. Z uwagi na jego duży potencjał, rozszerzalność oraz łatwość użycia, stał się on jednak oddzielnym projektem, używanym w dziedzinach o których autorzy początkowo nawet nie myśleli (studia fizyczne dla projektu Kolajdera Liniowego, czy precyzyjne testy algorytmu PHOTOS opisane w rozdziale 18).

Metodologia przeprowadzania testów zaimplementowana w narzędziu MC-TESTER została opracowana dużo wcześniej - w trakcie prac nad generatorem TAUOLA. Bazuje ona na porównaniach serii dystrybucji (histogramów) mas inwariantnych które da się zbudować ze zmiennych kinematycznych w fizycznym procesie rozpadu cząstek. Test taki okazał się mieć bardzo duży potencjał diagnostyczny wobec problemów, z jakimi typowo spotykają się autorzy generatorów: pozwala zlokalizować przyczyny niezgodności o fizycznym jak i programistycznym podłożu. Celami, które postawiliśmy przed narzędziem MC-TESTER były również prostota integracji środowiska testowego z istniejącymi środowiskami używanymi przez generatory i automatyzacja, zarówno procesu zbierania danych, jak i ich analizy, co w maksymalnym stopniu wyeliminować ma nie tylko nakład pracy. konieczny do zorganizowania testów, jak i możliwość popełnienia pomyłek.

Analiza prowadzona przy pomocy MC-TESTER'a składa się z dwu etapów: najpierw uruchamiane są dwa (lub więcej) porównywane generatory przypadków. Ich kody wymagają minimalnych zmian w celu wywołań procedur MC-TESTER'a . Po wyprodukowaniu każdego przypadku MC-TESTER przeprowadza analizę danych zawartych w strukturze event record, wyszukując procesy rozpadów wyspecyfikowanej

cząstki, klasyfikując kanały rozpadu i wypełniając histogramy wartościami odpowiednich mas inwariantnych. Jako wynik otrzymuje się pliki zawierające histogramy i informacje o zlokalizowanych kanałach rozpadu. W drugim etapie dwa pliki z histogramami są porównywane w środowisku analizującym dostarczanym przez MC-TESTER'a; w wyniku otrzymuje się "broszurę", zawierającą ogólne informacje na temat kanałów rozpadu zidentyfikowanych w obu generatorach oraz dla każdego kanału listę rysunków prezentujących odpowiadające sobie histogramy mas inwariantnych z obu generatorów. Każdy rysunek zawiera również dystrybucję przedstawiającą iloraz dwu (unormowanych) histogramów oraz wartość *SDP* ("*Shape Difference Parameter*") wyrażającą różnicę między histogramami w postaci liczby rzeczywistej, np. z przedziału $[0 \ldots 1]$. Podejście to pozwala przeprowadzać porównania histogramów które różnią się jedynie nieznacznie, względne różnice widoczne są w dystrybucji ilorazu oraz wartości parametru *SDP* (dla obliczenia którego, można użyć jednego z dostarczonych przez autorów algorytmu, eliminującego fluktuacje statystyczne lub dostarczyć własny algorytm).

Otrzymana broszura z analizą pozwala w prosty i szybki sposób zidentyfikować systematyczne rozbieżności w wynikach produkowanych przez dwie symulacje. Tabele podsumowujące wyniki pozwalają najpierw zidentyfikować kanały, w których występują rozbieżności, odsyłając następnie osobę prowadzącą testy do odpowiednich rysunków z dystrybucjami, gdzie "rzutem oka" ocenić można naturę i zakres rozbieżności. Na podstawie informacji uzyskanych z takiej wstępnej analizy można najczęściej od razu zidentyfikować źródło rozbieżności lub zawęzić rejon poszukiwań szczegółowych.

Dostępność narzędzia MC-TESTER oraz fakt, iż PHOTOS okazał się być stosowanym przez kolaboracje eksperymentalne do precyzyjnych symulacji, a także podobieństwo zawartych w nim algorytmów do tych stosowanych w generatorach typu *parton shower,* stała się motywacją do rozpoczęcia systematycznych studiów algorytmu PHOTOS w kontekstach jego fizycznej precyzji (w szczególności kwestia oceny błedu systematycznego) i potencjału użycia dla przyszłych generatorów QCD. Część tych, wciąż trwających studiów jest udokumentowana w szóstej części, będąc jednocześnie przykładem metodologii używanej w procesie walidacji symulacji fizycznej. Przedstawione wyniki są do pewnego stopnia kwintesencją pracy doktorskiej, w zakresie eksperymentalnej fizyki wysokich energii, z uwagi na ich zastosowania w eksperymentach takich jak Belle czy BaBar, obecnie zbierających i analizujących dane.

Przez blisko PHOTOS uważany był za symulację dającą jedynie przybliżone wyniki. Kwestia błędu systematycznego modelu fizycznego, który implementuje nie była istotna dla studiów i analiz wykonywanych w tamtym czasie - pogląd ten podzielali również autorzy. Porównania z danymi o dużej statystyce dla rozpadów bozonu W skłoniła autorów do wprowadzenia wagi korekcyjnej poprawiającej precyzję modułu dla tych procesów. Zainteresowanie kolaboracji eksperymentalnych badających fizykę kwarków b i c zmotywowała natomiast wprowadzenie uniwersalnej wagi interferencyjnej, której brak był wcześniej głównym źródłem niedoskonałości PHOTOSa. Jednocześnie rozwój techniczny algorytmu (eliminacja problemów numerycznych, wersja eksponencjonowana emisji wielofotonowej), opisane w sekcji 13.3 stały się zaczątkiem systematycznych studiów jego potencjału.

W przedstawianych tu testach skierowaliśmy naszą uwagę na precyzję PHOTOS'a w leptonowych rozpadach bozonów Z i W. Wybór tych procesów spowodowany był historią samego algorytmu, który powstał w wyniku studiów elementu macierzowego dla procesu $Z^0 \to \mu^+\mu^-$ oraz potrzebami eksperymentów na akceleratorach LHC i Tevatron. Proces ten stał się dlatego oczywistym kandydatem dla testów jakości aproksymacji użytych w PHOTOS'ie. Zdecydowaliśmy się na przeprowadzenie testów porównawczych z użyciem MC-TESTER'a, w których PHOTOS skonfronto-

wany mógł być z generatorami o wysokiej precyzji, takimi jak KORALZ i KKMC. Z uwagi na niejednoznaczności związane z regularyzacją podczerwonej rozbieżności w emisji fotonów w poszczególnych generatorach (ilość emitowanych fotonów, minimalna energia), konieczne było rozszerzenie metody używanej przez MC-TESTER'a (szczegóły w sekcji 18.1). Zarówno w testach na emisję pojedynczego fotonu, jak i w testach gdzie porównywane były mechanizmy emisji wielofotonowej ustalono, iż PHOTOS dostarcza wyników o precyzji lepszej niż 1 promil dla procesów rozpadów bozonów Z i W, zaś natura rozbieżności jest zrozumiała. Zidentyfikowano również potencjalnie interesujący efekt wskazujący na to, iż wielofotonowa wersja PHOTOS'a może reprodukować częściowo efekty rzędu $NNLO$ w rozpadach Z - w tym wypadku konieczna jest jednak bardziej skrupulatna analiza, przeprowadzona metodami formalnymi, nie zaś tylko symulacyjnymi. Zagadnienie to jest obecnie intensywnie studiowane i wykracza poza przedstawianą pracę.

Rozpoczęto też współpracę z kolaboracjami eksperymentalnymi zaangażowanymi w pomiary fizyki kwarku b i c (NA48, Belle, BaBar) nad studiami precyzji PHOTOS'a w rozpadach mezonów B, D, K. Bezpośrednim następstwem tych prac była implementacja uniwersalnej wagi interferencyjnej, poprawiającej znacząco precyzję PHOTOS'a dla tych zastosowań. Prace nad porównaniami PHOTOS'a z danymi eksperymentalnymi tych kolaboracji są obecnie w toku.

Jestem świadomy faktu, iż metodologia oparta na porównaniach wyników symulacji komputerowych, raczej niż na obliczeniach analitycznych, może rodzić nieufność w środowiskach fizyki teoretycznej. W dużej części podzielam opinię, iż ostatecznie przesłanki i wnioski, które udało się otrzymać przy pomocy prezentowanych testów, powinny być zweryfikowane na drodze skrupulatnych obliczeń matematycznych. Mimo że obecne zastosowania eksperymentalne nie wymagają obecnie tego typu precyzyjnych analiz, prace zostały rozpoczęte by zbadać potencjał algorytmu w przyszłych zastosowaniach dla QCD. Jednocześnie testy, które zostały przeprowadzone są ważną weryfikacją techniczną kodu PHOTOS'a, która ukazuje w jak dużym stopniu udało się w ostatnim czasie zapanować nad kłopotami z numeryczną stabilnością algorytmu. Wierzę, że metodologia oparta na testach modeli wyrażonych w postaci symulacji komputerowych, jest komplementarna do metod analitycznych, których jakości fizycznej zapewne nigdy nie zastąpi O wyborze metody decydować - w moim przekonaniu - powinien pragmatyzm.

Symulacje komputerowe będą miały kluczowe znaczenie w interpretacji danych pochodzących z eksperymentów na LHC. Przez wielu traktowane jedynie jako narzędzie ekspresji modeli w formie, która możliwa jest do rozwiązania przy pomocy komputerów, są - w mojej opinii - niedoceniane jako element metodologii nauki. Od strony technicznej przyciągać mogą one rzesze młodych "technokratów", widzących w nich jedynie techniczne wyzwania programowania dla celów nauki. Z perspektywy blisko 10 lat własnych doświadczeń, które zaowocowały przedstawianą pracą, skonstatować mogę kompleksowość problematyki. Wymaga ona beigłości w dziedzinach zarówno ścisłych: fizyka, matematyka, opanowania niezbędnych środków technicznych (inżynieria programowania, systemy operacyjne, ale i budowa detektorów), jak i ... elementów socjologii i komunikacji, niezbędnych w dyskusjach między zróżnicowanymi środowiskami zaangażowanymi w symulacje komputerowe dla fizyki wysokich energii.

Contents

Part I

Introduction

Chapter 1

Introduction

One of the central goals of Physics is to describe the observed phenomenons in a clear, formalized way, by means of rules, equations and laws. On one hand, observations and experiments provide us with their results, on the other, theoretical studies produce models, which try to describe the reality observed in the experiments. Understanding the

$$theory = experiment \tag{1.1}$$

equation becomes an essential part of our quest to understand the Nature.

In the High Energy Physics (HEP), due to the microscopic nature of the objects of the studies, this, somewhat philosophical, problem becomes yet more nontrivial, due to the effects predicted by Quantum Mechanics, such as the Einstein-Podolsky-Rosen paradox[1], which manifest often cross-intuitive and mysterious nature of our World.

To be able to put the equality relation in the equation (1.1), and make new discoveries, one needs to consider, in the data analysis, all of the effects of already known physics on the left-hand side of this equation. At the same time a very good understanding of the conditions in which the experiment in performed need to be assured (e.g. the precision of the measurements need to be known, the acceptance and the sources of background need to be understood, etc.). This often requires that the elements and the prediction taken from the theoretical model need to be embedded in the experiment design and data analysis. On the other hand, the free parameters of the model often cannot be determined from the first principles and need to be set to the values measured in the experiments. The equation (1.1) is entangled on both sides of the equality relation operator. In the philosophy of science, the term "phenomenology" is often used to describe the intermediate body of knowledge, between theory and experiment.

One of the most important factor that limits the potential for research and discovery in the high energy physics is complexity: theoretical descriptions, such as the Standard Model, or Supersymmetry, require the use of complex and abstract mathematical machinery of quantum field theory, group theory and theoretical mechanics. The experimental setups, needed to study the behaviour of the smallest constituents of the matter, are amongst the largest and most complex scientific apparatus ever built by human nature, and require energy concentrations that were present before only at the early stages of the Universe, capabilities for detection and tracking of thousands of microscopic objects that move with the speed of the light. The signatures of the new physics, which are going to be looked for by the experiments at the LHC, the largest particle accelerator which will soon be available at CERN are often very rare, and their signal-to-noise ratio is often very low - the detection of the Higgs boson (in one of its rare decay channels with clear signature)

is expected once every 10^{12} particle collisions and seems to be much more complex than finding a needle in a haystack. This require very sophisticated and efficient mechanism for data filtering (event selection or "trigger" systems), to retain only the data that contain potentially interesting physics signatures. Ultimately, due to mutual entanglement of theory and experiment, putting the equality operator in the equation (1.1) is only possible by means of complex computer simulations, that needs to make use of computer models of theory being tested and the simulations of the experimental setup. Matching subsequent steps of the large simulation chains, assuring that physical meaning and interpretation of data is preserved is a non-trivial task not only at the technical, but also at conceptual modeling level. Exploring phenomenological dependencies between the experiments adds another level of complexity.

A typical method to deal with the problems of high complexity is to find an approximated solution, based on the extrapolation of the studies of less-complex systems (system approach). The complex system needs to be decomposed to simpler parts, which are easier to understand and describe in simplified, semi-independent way, then the behaviour of the system might be determined using the simplified *models* of the components. The system approach is used generally to provide solutions to complex problems; for instance, in may be attributed to the solutions based on picture of interaction in quantum mechanics, since its early days. The approach is also particularly suitable for implementation in form of computer simulations (especially when Object-Oriented programming methods are used): the properties and interactions of the approximated models are expressed by modular computer code. Modular computer simulations based on system approach remain the only applicable method to confront the experimental results in high energy physics with predictions of theoretical models. Monte Carlo simulations are also the most widely used way to express the theoretical models (and conserve their predictions) of high energy physics.

In this dissertation I shall discuss computer simulations in the wide context of the phenomenology of the future LHC experiments. To get the broadest possible view, I dedicated my studies to two extremes: a few examples of the HEP Monte Carlo simulations, based on theoretical description of the processes on one side, and the `at2sim` computer model of the Dataflow System of the ATLAS detector at CERN, on the other side. The subjects in between these extremes, like the simulation of the detector response, or physical analyses, are outlined in less detailed way, only to expose their relations, significance and implications in the global view. It is the topic of a multitude of theses, such as [2, 3, 4] and published documents, hence I shall not repeat these informations here. The boundaries and the inter-play of the components of simulations, which is a usually neglected topic, are also discussed. From the practical point of view, the contents of this thesis document the tools I developed as a preparation step for the simulations for the LHC experiments, which may start experimental data-taking already next year.

In my studies I have chosen the ATLAS experiment as the example: it is the largest high-energy physics detector ever built, and probably the most complex physics experiment ever held. This choice was also motivated by the fact that the Cracow physics community is deeply involved in the ATLAS collaboration. The ATLAS experiment is also the topic of my own scientific and professional involvement (and interest) already for some time now; during the period 2001-2003 I participated in the development of the ATLAS Dataflow System in the EP/ATR group at CERN as CERN Doctoral Student. The part *II* of the thesis, which describes the `at2sim` model, is a documentation and report of my scientific activity from that period of time.

In years 1999-2001 and 2003-2006 I participated in the development of Monte

Carlo simulations for QED radiative corrections in particle decays. This thesis (in certain parts) is the documentation of the activities related to the PHOTOS Monte Carlo generator, and to a lesser degree, to the TAUOLA and TAUOLA Universal Interface, which triggered the need for creation of the MC-TESTER tool.

1.1 Main themes

The thesis approaches the main topic from a few sides. In general, it is concentrated on the phenomenology of the LHC experiments and my experience. From the experimental physics point of view, I will lead my discussion taking the ATLAS experiment as the example, however I will also refer to certain aspects of other experiments, like Belle, BaBar or Linear Collider, which appeared in a natural and unavoidable way in my considerations as parallel applications or as a source of phenomenological input.

The part of theoretical physics which is discussed in the thesis covers radiative corrections in particle decays, as implemented in the PHOTOS algorithm, and also some aspects of the phenomenology of the τ-lepton decays. The choice of PHOTOS and QED radiative corrections was made to explore in more detail one of a functional elements of the large computer simulation chain, and to observe the dependencies and relations with the other parts. The PHOTOS algorithm is well suited for this purpose: it explores the subtle points of data exchange and employs a variety of algorithms, while being contained in a small block of computer code.

The software-engineering aspect of the thesis is the construction and matching of various algorithms and data structures used in high-energy physics simulations. This aspect is particularly interesting because of the fact that the simulation software for the LHC will need to employ smaller functional packages, typically written in languages other than (adopted by the LHC experimental collaborations) C++ (i.e. FORTRAN – C++ mixing bridges will certainly be needed to enable existing generators to be used in the collaboration software) and data structures.

The use of components, which obviously has advantages and limitations, will be one of the common topics discussed in various contexts in this thesis. Especially, the difficult topic of component definition (and re-definition) in context of increasing physics precision will be discussed.

Ultimately, a more technical, experiment-related part of this thesis is devoted to computer modeling of the ATLAS Dataflow System. The complete simulation project, starting from the problem analysis and studies of the behaviour of the components of the system, through the development, parameterisation and validation of the simulation models, to the final result, is described. This part also documents my practical experience in organising and conducting experiments and measurements, preparing experimental setup, developing own tools and using them, data analysis and interpretation. This vital element of scientific methodology had to be mastered before actual simulation model could be formulated.

1.2 Outline of the thesis

The thesis is organised in seven parts, each of which is subdivided to chapters that are dedicated to discussion of individual topics.

Part *I* of the thesis serves as a general introduction to the main topic. It presents the general view on the computer simulation methodology and brief descriptions of general topics related to their use.

- In Chapter 2 I discuss computer simulations as a part of contemporary scientific methodology. The use of system approach to obtain solutions for complex

models, combined with computer-based methods lead to computer simulations (modelling) methods, which are considered the third main pillars of scientific methodology, peer to theory and experiment. A general classification of solution methods for physics-related simulations is given, and the keywords, such as *phase space,* are explained. A history of computer simulations is also shortly presented, showing their deep relations with nuclear physics, from the very beginning. The potential of the computer simulation methodology, in context of their use in experimental high energy physics, is presented. Ultimately, in section 2.7, the methodology for the use of computer simulation as a scientific tool is presented. This section is the quintessence of the chapter and presents the methodology I worked out on the basis of my own practise and various sources in literature, that discuss scientific method generalities. The specification of this methodology is the first of my research achievement presented in the thesis. It provides solid scientific foundation to the other achievements related to computer simulations, presented in the next chapters. Chapter 2 is finished by the discussion of the importance of validation in the methodology, presentation of elements: algorithms, data structures and tools, and the discussion of the applicability of the software development process approaches, as used for the development of general-purpose software, to scientific simulations.

- In Chapter 3 I give an overview of the use of computer simulations in high energy physics, in particular I discuss their role as a design tool, and their role in experimental data analysis and interpretation; for the latter I give a description of the elements of "full chain simulation" on the example of ATLAS. I close this chapter by a discussion of physics being embedded and conserved in form of computer (typically: Monte Carlo simulation) codes

- In Chapter 4 I present the component-based approach to computer simulations. Despite the method being widely used for general-purpose software, its adoption in computer simulations was not immediate. The discussions of this topic in context of discrete-event simulations may easily be found in articles and conferences related to scientific simulations, yet their use in the area of high energy physics does not seem to be present. The overall concepts and keywords are presented at first. Then, the discussion of the use of components in the simulations for high energy physics is provided, pointing out the use of architectures in the off-line analysis software, the deficiencies, and the lack of adoption of this important technology in the area of HEP Monte Carlo event generators. One of the most important issues: the definition of the scope of the components, depending on the required physics precision is also discussed.

- In Chapter 5 I give an overview of the ATLAS experiment. The physics program, requirements and the layout of the detector provide a source of general, reference information. Then, the ATLAS Trigger and Data Acquisition system is shortly presented. The layout of the ATLAS Data Flow system is discussed in more details - the performance of this subsystem is the topic of my studies (involving the `at2sim` simulation) presented in part *II*.

Part *II* of the thesis describes my research activities in the domain of the ATLAS Dataflow System and documents my achievements from the period of CERN Doctoral Student schollarship, in years 2001-2003. The main topic of these studies was the performance of the ATLAS Dataflow system. The `at2sim` computer model was created to predict the behaviour of the full-scale system and extrapolate the results of measurements performed on small "test-bed" laboratory setups. The simulations were preceded by in-depth study of the behaviour and performance

of the components of the system, namely: the (Ethernet-based) networks and the computer-based nodes running Linux operating system. This required the creation of own software tools for network performance measurements, understanding the mechanisms of the Linux kernel related to networking, experimenting with Linux tunable parameters to optimise the performance. It also involved the use of the ATLAS Dataflow (Data Collection) Software, developing the methods for performance measurements and good understanding of all its components.

- Chapter 6 presents the motivation for the `at2sim` simulation, its history and the most important facts that lead to most important design decisions.
- In Chapter 7 I present the studies of performance of networking in Linux and ATLAS Data Collection software; the issues related to the use of interrupt mitigation techniques, which seem to be essential for data transfer at Gigabit Ethernet network links are exposed. Testing tools, experimental setups, measurements and analysis of obtained results are provided. This step makes foundation for the design of the `at2sim` model, and provides the initial values for `at2sim`'s parameters.
- In Chapter 8 the Ptolemy package used as the architecture for the `at2sim` model is presented. The discrete event modeling approach is also explained.
- In Chapter 9 I present the `at2sim` model. The models of system components, with their parameters are briefly described - all models of the applications (computer-based nodes) are based on the generic model of a networked, Linux-based computer, the model of which is presented in more details. The process of validation, based on comparison of model results with the models of "testbench" experimental setups available at the laboratory is then presented. The chapter is closed by presentation of results of modeling and message-passing studies, that formed important milestone in preparation of the ATLAS T/DAQ and DCS Technical Design Report document.

Part *III* of the thesis presents the discussion and my considerations related to the subject of data structures for HEP Monte Carlo simulations, which seems to be - at technical level - one of fundamental, unresolved problems for HEP Monte Carlo simulations.

- Chapter 10 presents general introduction to the concept of event record, the requirements and scopes of use. It also describes general problems related to this approach.
- Chapter 11 gives an overview of event record standards use in FORTRAN and a few C++–based event generators. It also presents my own experiments in that area, namely the `HEPEventLib` unification layer for FORTRAN-based event record standards, which was then used successfully in the `MC-TESTER` tool. A few conclusions and outlook comments, questioning the validity of the approach based on common event record for the whole simulation chain, finish this chapter.

In part *IV* I present the examples of HEP Monte Carlo event generators, `PHOTOS` and `TAUOLA`, that gives a good overview on the methods and problems of interfacing functional modules into large simulation chains. The two project present two extremely different approaches (completely "abstract" interface for `TAUOLA`, the `HEPEVT` event record for `PHOTOS`).

- In Chapter 12 I present a short overview of Monte Carlo event generators.

- Chapter 13 describes in more details the PHOTOS Monte Carlo generator for QED radiative corrections in particle decays. I was personally involved in the development of recent improvements of the algorithm. This chapter may also serve as a part of technical documentation of the PHOTOS algorithm

- In Chapter 14 TAUOLA and its Universal Interface are briefly discussed. The algorithm and underlying physical models are not described (they are discussed thoroughly in other, recent theses: [2, 3]); instead, technicalities of the interface (that were not described previously in enough detail) are presented. The issues related to the use of TAUOLA in experimental collaboration, and maintaining of the versions of code is also signalled, and left for further discussions in Chapter 16.

Part *V* provies the description of the tools I employed and developed myself for the purposes of design, development, testing, validation and maintenance of the simulations.

- In Chapter 15 I shortly present the variety of the tools related to code development. Large C++ simulations developed nowadays require the use of more advanced tools for code management, build system infrastructure or third-party libraries.

- Chapter 16 documents the TAUOLA-PHOTOS-F environment for maintaining the versions of PHOTOS and TAUOLA generators. The versions of the source codes as used by various experimental collaborations may be reconstructed, and bound to one of specified interfaces. The creation of a stable version-management package, preserving the physical content of the generators, is an essential step towards future translation of these generators to C++.

- In Chapter 17 I present MC-TESTER: a universal tool that performs semi-automatic comparison tests of Monte Carlo event generators. The tool was initially created as a part of the TAUOLA-PHOTOS-F environment, yet due to its flexibility it evolved to an independent project. It was successfully used in various environments, helping to find the errors in large collaboration software chains. It was also essential in the development of new, experimental versions of the TAUOLA package, for *Belle* and *BaBar* collaborations. Together with the TAUOLA-PHOTOS-F it forms a ready for use environment prepared for re-implementation of the TAUOLA generator to C++. It was also the main tool used in precision tests of the PHOTOS algorithm, described in Chapter 18. Initially, it was a surprise that MC-TESTER might find the use outside the TAUOLA-PHOTOS-F, from which it originates; it's availability made the precision tests (and recent improvements) of PHOTOS possible.

In part *VI* (Chapter 18) I present the study of precision of the PHOTOS Monte Carlo library. This is the most important achievement of this thesis, from physical point of view, because of present applications in *Belle* and *BaBar* experiments. By use of tools, techniques and methodologies presented in the thesis, the physical precision of the PHOTOS algorithm was firstly enhanced, then, by means of the MC-TESTER tests, it was demonstrated that this universal, relatively compact and fast Monte Carlo generator, always thought to be a low-precision tool, may in fact provide high-precision predictions (for certain channels that were studied), that agree to per-mile level with large generators that employ Matrix Element calculations. The studies of the potential of the algorithm for future applications, which are not described in this thesis, have already begun [5, 6]!

A summary and conclusions are presented in Part *VII*, that closes the thesis.

Chapter 2

Models, simulations and scientific method

In this chapter I discuss the place of computer simulations in scientific methodology. The "traditional" method, which is described briefly in the next subsections, was extended in the last decades, to include the tools and methods provided by computer modeling in its methodology.

The topic of scientific methodology is covered in a variety of books and articles. For the purpose of this dissertation I referred to the material found in [7]; in particular, for the purpose of the discussion in this chapter I follow the description and terminology concerning the traditional scientific method given in [8].

2.1 Definition of a scientific method

Science is the process of studying, discovering and describing the Laws of the Nature. This process may be influenced by personal or cultural beliefs of the researcher, and affect the perception of natural phenomenas and their interpretation. A scientific method is the use of standard procedures and criteria in order to minimise those influences when developing a scientific theory. In other words: a scientific method attempts to minimise the influence of bias or prejudice of the experimenter when testing an hypothesis or a theory.

There is no single, universal scientific method: instead of "The Method", scientists commonly use agreed methods, designed to be functional and to achieve the goal of delivering a theory which "is true".

2.1.1 Traditional scientific method

The traditional approach to the scientific method is composed of four steps:

1. observation and description of a phenomenon (or a group of phenomena),
2. hypothesis formulation. In physics, the hypothesis often takes the form of a causal mechanism or a mathematical relation,
3. use of the hypothesis to predict the existence of other phenomena, or to predict quantitatively the results of new observations,
4. experimental test of the prediction in properly performed experiments performed by several independent experimenters.

The result of the test may bear out the hypothesis, so it may come to be regarded as theory or law of Nature. If the experiments do not bear out the hypothesis, it must be rejected or modified.

Unlike a mathematical proof, a "proved" scientific theory is always open to *falsification* if new evidence is presented. Even the most basic and fundamental theories may turn out to be imperfect if new observations are inconsistent with them. Further, no matter how elegant a theory is, its predictions must agree with experimental results if we are to believe that it is a valid description of nature. In physics, as in every experimental science, "experiment is supreme" and experimental verification of hypothetical predictions is absolutely necessary.

In physics and other science disciplines, the words "*hypothesis*," "*model*," "*theory*" and "*law*" have different connotations in relation to the stage of acceptance or knowledge about a group of phenomena. A *hypothesis* is a limited statement regarding cause and effect in specific situations; it also refers to our state of knowledge before experimental work has been performed and perhaps even before new phenomena have been predicted. The word *model* is reserved for situations when it is known that the hypothesis has at least limited validity (a often-cited example of this is the Bohr model of the atom). A *scientific theory* or *law* represents a hypothesis, or a group of related hypotheses, which has been confirmed through repeated experimental tests. Theories in physics are often formulated in terms of a few concepts and equations, which are identified with "laws of Nature," suggesting their universal applicability.

2.2 Models

The concept of the model is quite general: different types of models are commonly used in practise, such as physical (scale-down) models, graphical models or mathematical models. In this subsection let us discuss in more detail the role of the model in scientific methodology.

We start with the definition of the model - the shortest possible description, given in [9] states: “a model is an abstract description of a system”. Such definition is however quite limited and does not provide the classification of models, and their context in the scientific method.

The most abstract and elementary is the *conceptual model*. Let us quote the description from Wikipedia[1]:

> “Conceptual (or abstract) model is a theoretical construct that represents physical, biological or social processes, with a set of variables and a set of logical and quantitative relationships between them. Models in this sense are constructed to enable reasoning within an idealised logical framework about these processes and are an important component of scientific theories. (...)

[1] Wikipedia [10], the web-based encyclopedia written collaboratively by volunteers, becomes more and more popular source of information nowadays. Despite the few incidents, reported widely by media, of abuse of its rules and introducing false information, it still remains one of the most reliable, freely-accessible, comprehensive source of information used by young generation of scientists nowaday, who use the Internet as one of their research tools. This group of the Internet users is (usually) aware of dangers related to the openness, accessibility and freedom of publication of any content on the web . Because of its popularity in the scientific world, the information delivered by Wikipedia present wide range of points of view, providing the reader with a clear overview of discussed topics, and encouraging further quest for additional sources of information. I believe that the use of Wikipedia as a source of such overview information, with appropriate distance and awareness of possible sources of mis-information, is justified for the purposes of this dissertation. According to the Nature magazine [11], Wikipedia' competence in describing scientific topics is comparable to the one of renowned *Encyclopedia Britannica*.

> The purpose of a model is to provide an argumentative framework for applying logic and mathematics that can be independently evaluated (for example by testing) and that can be applied for reasoning in a range of situations. Models are used throughout the natural and social sciences, psychology and the philosophy of science.
>
> Abstract models are used primarily as a reusable tool for discovering new facts, for providing systematic logical arguments as explicative or pedagogical aids, for evaluating hypotheses theoretically, and for devising experimental procedures to test them. (...)
>
> A conceptual model is a representation of some phenomenon by logical and mathematical objects such as functions, relations, tables, stochastic processes, formulae, axiom systems, rules of inference etc. A conceptual model has an ontology, that is the set of expressions in the model which are intended to denote some aspect of the modeled object. (...)"

Following point 3 in the scientific method in Section 2.1.1, the model should be used to obtain predictions, which in turn should be confronted with observations and used to study the behaviour of the modelled system. This requires that the *solution* of the model is obtained.

The use of models is widespread in many branches of science and engineering - most commonly the term is used to refer to scale models, which are representations of a real object, which is either smaller or larger in size than the object being represented. Scale models are often used to guide the construction of the final object at the full size: testing the performance of the designed object may be performed without the need for building a full-sized prototype; mechanical engineering and hydrodynamics are the domains where the scale model are particularly useful. Collecting scale models of cars, airplanes, etc are also one of the most popular hobbies...

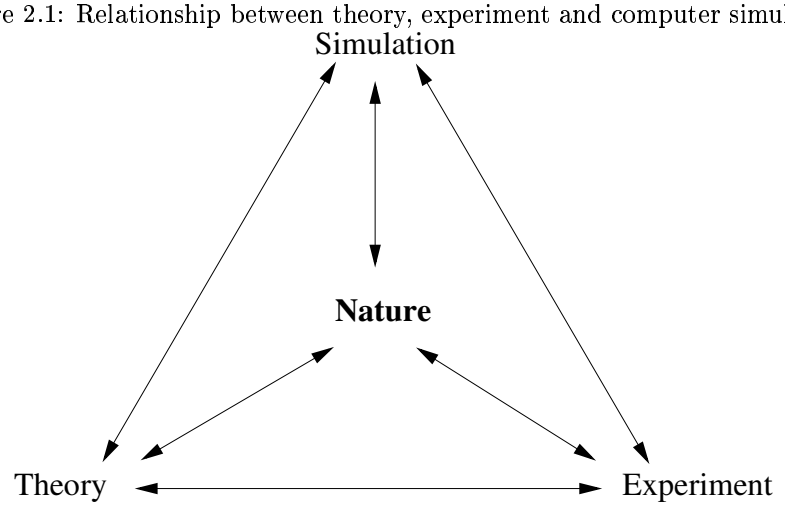
2.2.1 Models of physical system: states, phase space

In modeling of physical systems, the aim is to calculate the observables and dynamic of the modelled system. Typically, the values of the properties are calculated as averages on a sample of some parameter-space. For that, the *state* of the system needs to be defined using *state variables*, as $\mathbf{x} = (x_1, \ldots, x_n)$, where n is the number of degrees of freedom. The complete set of the states with a measure defines the *phase space,* Ω. The values of observables are calculated using, for instance, the rules of statistical mechanics as a function of states of the system by calculating the value along a path (trajectory) in the phase space. The way in which the trajectory is chosen classifies the model (and the method of its solution) to be either deterministic or stochastic.

2.2.2 Classical methods for model solution

The conceptual models expressed in form of mathematical equations or objects are often refered to as mathematical models. Typically, differential equations are used for continuous systems, and difference equations for discrete ones. In traditional methodology, the exact or approximated solution to mathematical model was obtained by analytic transformations of mathematical formulas in order to obtain the expressions for state variables or output variables as functions of time and system behaviour. Such expressions could then be evaluated to obtain the predictions for any value of input variables, within the range of model validity.

Figure 2.1: Relationship between theory, experiment and computer simulation.



The availability of computers made it possible to obtain approximated numerical solutions for more and more complex models, including the ones for which the solution in analytic form does not exist (e.g. approximated solutions for non-linear partial differential equations). Despite the fact that the calculations were performed by computers, the actual methodology used was the "classical" one - no computer simulation was involved yet.

2.3 Computer models and simulations

The use of computers to obtain numerical solutions certainly increased the number of classes of models that could be solved and studied. Nevertheless, certain classes of models, in particular the ones of large complexity due to the number of elements, still couldn't be solved using "traditional" method. For such cases, where expressing the model in form of mathematical formulae is not possible, or various methods need to be combined, *computer simulation modeling* may provide a viable way of obtaining the solution and predictions.

A computer (simulation) model is a conceptual model implemented on a computer. The experiments performed on such model are called simulations. They typically involve evaluations using numeric techniques, often carried out with dedicated software tools, aided with visualisation and data analysis programs.

Computer models and simulations quickly became very popular everywhere in science: in economics, biology, engineering, physics. The use of computer simulations and the validity of various simulation methodologies are subjects of debate in the philosophy and methodology of science. Nowadays computer simulations and modeling have been accepted as new, integral parts of scientific methodology and claimed (i.e. by Ken Wilson, a Nobel Prize laureate) the third paradigm of science, after observation and theory. Following the description in [12], I present this fact in Figure 2.1.

The authors of [12] also argue that the goal for the simulations is not to simply provide "a better fit" to experimental data, rather than does analytic theory. The goal is to better understand the physical properties and processes, taking advantage of the possibility of performing a perfectly controlled computer experiment, where all the aspects of the system configuration may be explored in detail.

Simulation modeling can be used not only to predict the behaviour of and existing system subject to some specified condition, but also to study a system even

before it is built (i.e. at its design phase), and to gain better understanding of the behaviour of the real system. One of the most important advantages of simulation models is that they do not have restrictive simplifying mathematical assumptions, common to analytical methods, hence almost any performance measure can be estimated from data generated with simulation models. In some cases, simulation may be the only approach that can be used to reach a solution to a given problem, in some others - analytic solutions will remain superior.

2.4 Simulation methods

Following the classification schemes in [13] and [9], the methods (or techniques) of computer simulations may be categorised as stochastic and deterministic.

In deterministic methods, all state variables follow a deterministic law, usually expressed by a mathematical function,therefore the behaviour of system is entirely predictable. In physical models the trajectory in the phase space is determined by internal dynamics of the system: equations of motions are integrated in time. For instance, for a set of particles in classical mechanics, this leads to the trajectory $(\mathbf{x}^N(t), \mathbf{p}^N(t))$ in the phase space, with boundary conditions given by positions $\mathbf{x}_1(0), \ldots, \mathbf{x}_N(0)$ and momenta $\mathbf{p}_1(0), \ldots, \mathbf{p}_N(0)$. The most popular deterministic method is Molecular Dynamics (MD). The starting point for MD is a well-defined microscopic description of a physical system (i.e. the system of one or multiple bodies, with Hamiltonian or Lagrangian equations or Newtonian equations of motion. In MD, the statical and dynamical properties of the system are calculated based on the equations of motion - the equations are solved using classical, numerical methods (quadratures) based on approximation of differential operators by finite-size differences and discrete variables. The equations of motion may, in general, be given as stochastic differential equation, which leads to non-deterministic dynamics. Thermodynamical parameters of the system may be calculated by average over trajectory.

Stochastic models describe systems where (all or some of) the state variables in a way that is not completely predictable (these are called stochastic variables). The transitions from one configuration to another, which was given by momenta in deterministic methods, are realised in probabilistic evolution of Markovian Process (Markovian Process is a probabilistic equivalent of Internal Dynamics).

The simplest example of stochastic methods are Random Walk methods, which may be used e.g. for iterative solution of equations (Von Neumann Ulam method) of solving Laplace differential equation[14]. One of the hybrid method is Brownian Dynamics (BD), in which some degrees of freedom are represented by their stochastic impact on other degrees of freedom. The trajectory in the phase space is calculated for a set of molecules, which follow Langevin equations in force field.

The most widespread, however, is the Monte Carlo method (MC), which gives the solution of the problem in form of a parameter of some hypothetical population by using random sequence of numbers to generate a sample of this population; based on this sample, statistical estimation of the value of the parameter may be calculated. The scope of use of Monte Carlo methods in physics is indeed very wide: it may be for instance applied to statistical mechanics or various areas of quantum mechanics (QMC, Quantum Monte Carlo methods: QMC-DMC (diffusion MC [15]) for solution of Schrodinger equation, Path Integral Monte Carlo, etc). I am not going to describe general Monte Carlo techniques and methods here - they are covered in many books and articles. Instead, I will concentrate on the specific class of Monte Carlo simulations for high-energy physics.

Simulation models may also be classified to be either continuous models (which

change their state continuously with time, such as models based an equation of motion), and discrete-event models, which change their state at discrete points in time, as a result of the occurrence of events. An event, then, is an instantaneous occurrence that affects the state of the system. The at2sim model, discussed in part *II*, is an example of discrete event model.

From the point of view of the organisation of the simulation, certain popular methods are often refereed to as belonging to the same class; among the commonly used are particle methods and continuum physics methods (the finite difference and finite element methods).

2.4.1 System approach

"Abstraction is the most powerful tool available to deal with complexity"
[Dahl, Dijkstra, Hoare "Structured Programming", London, Academic Press 1972]

For modeling of the systems with large complexity, system approach is usually chosen: the quantities are calculated in sub-systems to be later used as parameters for the calculation of the complete system.

This approach, used extensively in statistical physics, may be generalised to model other systems of large complexity. The model, then, is composed of components, also called entities or objects, the properties of which depend on their attributes. The attributes of the objects are altered as a result of (inter-)actions and operations. The values of the attributes taken to describe the system at any instant of time determine the state of the system. In a conceptual model, this set of attributes is represented as a set of variables called *state variables* - their values at any given instant determine the state of the system. The concept of state variables is the term well known in theoretical and quantum mechanics. It is also widespread in Object-Oriented programming as data encapsulation and data hiding techniques.

2.4.2 Errors and uncertainties

Similarly to results obtained from measurements or analytical calculations (conceptual or mathematical models), the results obtained from computer simulations bear the errors and uncertainties. For computer models, new categories of uncertainties and errors are present: numerical, modeling and programming errors.

Programming errors are caused by programmer's mistakes in the model's computer code, and cause the simulation not to perform in the intended way. Because of necessary "human factor" - the programmer - they are inevitable, but (from the technical point of view) always possible to correct or eliminate, once they are detected. Majority of them (usually the trivial ones) may be detected by compilers, dedicated code verification tools (such as lint or ftncheck), or run-time debugging and testing systems (automated ones, such as valgrind, or implemented by the programmer inside the model). The most difficult to identify are the errors in the code that give reasonable-looking, but incorrect results or seem to impose additional constraints on the model. A bug of that type existed in `PHOTOS` for many years, and only recently was corrected: the results produced by `PHOTOS` were correct for single- and double-photon modes of generation, yet prevented the algorithm from generating larger number of photons (see also Chapter 13).

Numerical errors are induced by the necessity to use the numerical approximations rather than analytical formulae. This category of errors include the round-off errors, which are caused by finite (limited) precision of word size of the computer

and the truncation errors are due to the insufficient expansions of analytic expressions in numerical forms. Despite the fact that nature of these errors is well known (may be described in mathematical way), and the calculations are performed on high-precision variables now, they may not be neglected. In particular, the artifacts of such errors are still visible in older codes, which (in order to speed-up calculations performed on computers at that time) performed calculations using variables of limited precision (i.e. REAL*4 variables of FORTRAN). In `PHOTOS`, a special routine that compensates for numerical errors needed to be implemented (see Section 13.3.1 for details).

Modeling errors are the results of incomplete representation of the model, compared to that of the complete physical reality. This class of error is not strictly bound to computer models, but rather more general to modeling - it usually originates from the conceptual model. Modeling errors manifest themselves in insufficient precision of the predictions of the model, confronted with observations or measurements. This kind of errors cannot easily be eliminated: usually a new model, with higher precision, needs to be proposed, implemented and tested. Sometimes, parameters that approximate the uncertainties may be introduced to the model - by variating their values one may conclude the order of magnitude of uncertainties or errors (e.g. estimate the systematic errors of measurements, related to approximations used in theoretical model used to describe the data).

2.5 Potential of computer simulation modeling

As already outlined above, the main advantage of computer simulation was the ability to solve the models that could not be solved otherwise. In particular, harnessing system approach to deal with complexity, and the possibility of combining various methods are amongst the most spectacular demonstrations of its power. Let us now present other arguments that demonstrate its advantages.

Experiments, measurements and observations are more and more often performed using electronic equipment, and the data is obtained in digital, computerized form. Certain observations (such as high energy physics or astrophysics) cannot be performed in other way in fact. Computer simulations allow to verify new theories and models by comparing their predictions almost "directly" with experimental data, and asses the choice of particular model. Such confrontation of theory with experiment, which usually requires taking into account the imperfection of experimental setup (such as detector acceptance and geometry) would not be possible with other (analytic) techniques, because of increasing complexity of analysis.

Computer simulations also play important role in the design and optimisation of new experimental setup. The models of large experimental setups (such as HEP experiments) are usually much easier (and cheaper) to construct. Then, the models may be used to demonstrate the viability of the proposed experiment, and also optimise it (i.e. increase its discovery potential using predictions of various theoretical models).

Simulations are also often used as a priceless tool in designing large, complex systems, for which full-scale prototype cannot be built. In such case, properly designed, calibrated and validated computer model may be used to determine the behaviour of the system with full complexity, based on the observations of scaled-down prototypes. This possibility was exploited by the `at2sim` model, described in more detail in Chapter 6, to explore the behaviour and validate the design of the Dataflow System of the ATLAS experiment.

The availability of more and more powerful computers and simulation methods opened new areas of science, on the frontier of mathematics and physics - spectacular studies of computational complexity became possible, and allowed the exploration

of fractals, cellular automata's[16] and the theory of chaos.

The methods based on computer simulations may also fill the gap between theory and experiment: certain parameters or dynamics of systems are sometimes hard or even impossible to measure in real-world experiments, yet they may be calculated using computer simulations (e.g. percolation threshold[17]).

Of course, simulation methods have their limitations as well (such as the aspects related to analytical properties of results from fitted theories, which cannot be studied). The other, more serious, problem of the computer simulation modeling is of sociological type. Increasing complexity of the models, and the availability of simulation environments presenting the objects as easy-to-use "black-boxes" result in possibility of building and running simulations without any knowledge of underlying conceptual model. Certainly, this may lead to constraints and parameter validity regions of the models being neglected, and to situations where the predictions of the models are not correct or not sensible at all. The possibility of bugs in the simulations' codes aggravates the problem yet more.

2.6 History of computer simulations

In this subsection let me stop the, quite concise, presentation of computer simulation as a scientific method, to presents its history, which is tightly bound to physics. This description is far from being a complete time-line of the methodology - it is rather a set of highlights of the events that established the place of computer simulations in the high energy physics.

The history of computer simulations in physics began in 1945, and coincides with the origins of the first electronic computer, ENIAC, built during the wartime by a team of scientists, engineers and technicians at the University of Pennsylvania in Philadelphia. Initially the ENIAC, constructed as a military-sponsored project, aimed at automated calculation of firing tables. At that time the "Manhattan Project" was deployed to investigate the possibility of the construction of the new, enormously powerful weapon - thermonuclear bomb... For physicists parttaking in the project the task was to find a way to calculate the neutron transport equations for a spherical sample of material. In Spring 1945, John von Neumann suggested preparing the computational model of the thermonuclear reaction for the ENIAC. In Spring 1947 Polish physicist, Stanisław Ulam, who joined the team of physicists led by von Neumann, proposed the use of statistical techniques to solve the computational problem of neutron fissionable material. At that time N. Metropolis, who worked together with Ulam, suggested the "obvious name" for the statistical method - a suggestion not unrelated to the fact that Ulam's uncle used to borrow money from relative because he "just had to go to Monte Carlo" [18]. The Monte Carlo method was born [19].

Since then, computer simulation methods became a well-established and recognised tool in many branches of science [17].

In high energy physics computer simulations started to be in regular use already in 1970'ies [20, 21, 22]. The detector simulation package `GEANT` became available in 1978 [23]. However, Monte Carlo techniques based on mechanical "wheel of chance" was used by Wilson already in 1950'es:

> "(...)The procedure used was a simple graphical and mechanical one. The distance into lead was broken into intervals of one fifth of a radiation length (about one mm). The electrons or photons were followed through successive intervals and their fate in passing through a given interval was decided by spinning a wheel of chance; the fate being read from one of a family of curves drawn on a cylinder.

> A word about the wheel of chance: The cylinder, 4 in. outside diameter by 12 in. long, is driven by a high speed motor geared down by a ratio 20 to 1. The motor armature is heavier than the cylinder and determines where the cylinder stops. The motor was observed to stop at random and, in so far as the cylinder is concerned, its randomness is multiplied by the gear ratio. (...)"
>
> [R. R. Wilson, "Monte Carlo Study of Shower Production", *Phys. Rev.* **86**, 261 (1952)]
> cited already in 1978(!) in [24]

One of the most important element needed to perform calculations in high energy physics in the method for integration of multiple-body phase space. At CERN a computer code performing this task using Monte Carlo method was written already in 1960's: OWL (by Gerry Lynch), rewritten later by F. James as `FOWL` (W505) [25], then (in 1975) adapted as `GENBOD` (W515), may still be found in the `CERNLIB` [26] - the library of physics computer codes at CERN! The parts of the `FOWL` algorithm (or its adaptations) are still present in many event generators nowadays (e.g. in `TAUOLA`[27, 28, 29]). The method used in `FOWL`, dating back in 1968 is described in detail in [30] - let us shortly repeat the timeline described there. The history of `FOWL` and phase-space integrator seem to go back as far as to Fermi's studies of phase space integration in 1950, and Block's exact calculations in 1956. The first to apply Monte Carlo methods was Kopylov, in 1958, who manually created the list of two hundred "events". It was however after the discovery of factorisation property of the phase-space Lorentz-invariant formula by Srivastava and Sudarshan in 1958 that enabled the use of the method on computers: so called T-generators and M-generators, the first authors of which were probably Lynch, Goldhaber, Raubold and Kopylov, around 1960/61. Lynch incorporated the method in his OWL program for the CERN 7090 machine; this same code is used in `FOWL`.

2.7 Computer simulation development stages

In Subsection 2.1.1 I presented general scientific method as a sequence of steps. Let us now formulate a similar sequence of steps for development of scientific computer simulation model. This recipe should not be treated as the only valid formulation, and does not need to be followed strictly, on a step-by-step basis. A multitude of similar recipes, which in general are as good as the one described here, can be found in many sources. They differ slightly in details, but all of them contain common requirements, such as definition of the model, or validation. The recipe presented below is based on my own experience and also on recipe presented in [9].

General steps in computer simulation modeling

1. *Definition of the problem statement for the simulation model*
   This includes specification of the purpose of the simulation , questions it should help to answer, the performance and precision measures.

2. *Specification of the (conceptual) model*
   At this step a clear description of what is to be accomplished should be prepared. It should contain the list of relevant components, interactions and relationships among the components and the dynamic behaviour of the model. The conceptual model, including mathematical or physical formulae should be available.

3. *Computer model design*
   This involves establishing the data structures and the details of the algorithms.

4. *Data collection*
   At this step various type of data used to validate (or calibrate) the simulation model should be collected. This may require designing and performing experiments e.g. on simplified setups, and processing of the results of such measurements to extract the values of the model parameters.

5. *Model programming*
   This involves the implementation of the model in a programming language, or creating it in computer modelling environment (e.g. by means of graphical user environments).

6. *Verification of the computer model*
   This is a debugging stage, where the results produced by the model program are verified to match with those that would be produced by a correct model.

7. *Calibration/tuning*
   Involves specifying and tuning of the parameters of the model. At this step the data from the Data Collection step is used directly or indirectly. The values of parameters may also be tuned to obtain the best fit of the model to reference results.

8. *Validation of the model*
   At this step comparison of the output of verified model with the outputs of a real system ("reality check") is performed. Validation is discussed more thoroughly, for instance, in [31].

9. Steps 7 and 8 (tuning and validation) should be performed until certain level of confidence in model is gained, and the optimal values for tunable parameters of the models are found.

10. *Preparation and running the model (simulation)*
    collecting, processing and analysis of the results (predictions) of the simulation. Analyze the uncertainties of the results, impact of the approximations used in the model and the impact of other effects, such as limited statistics of the run.

In simulations of the complex system built of components, the steps specified in this general recipe require additional considerations. In such approach, the model of the complete system is built of a number of components, each of which may be a separate model, developed independently from the others. The way in which the decomposition of the system into components may be performed is one of the critical decisions, that need to be taken quite early. The decomposition and design of model components is usually determined by a balance between precision of the results and the availability of computing resources (CPU time, memory and disk space) and available manpower. Additional constraints on possible decomposition schemes may also be added by e.g. laws of physics. To deal with this problem, various approaches are in use. Among them, preparing exchangeable simulation components with the same functionality, with various level of complexity/precision, from detailed ones to simplified, parameterised. The use of parameterised, simplified models, such as `ATLFAST` [32], is particularly appealing: new models and hypothesis may quickly be evaluated in a rough way, before full, time-consuming simulation is executed to study the interesting problems (identified by fast simulations). The topic of component-based simulations is covered further in Chapter 4.

2.8 Scientific simulations and software development process

Increased complexity of the computer simulation models, demands that more and more advanced software development technologies are used. For instance, the codes of HEP Monte Carlo event generators count tens of thousands of lines (`HERWIG` 6.4[33]: 55 000 lines, `PYTHIA` 6.2[34]: 60 000 lines of FORTRAN code) and hundreds of classes (`ThePEG`, `ROOT`). Complete simulation software chains for the LHC experiments are composed of tens to hundreds of such packages. Managing software project of such complexity becomes a serious task, in particular in scientific environment. Even though projects of much larger complexity are successfully managed for years of their existence in commercial and open source worlds, using the schemes of so called software development process, they cannot automatically be adopted in all scientific projects. In particular, in HEP experimental collaborations, certain rules may become hard, impractical or even impossible to enforce.

The discussions over the applicability of the software development process to physical (or scientific) software only aggravate the antagonism between physicists and software development experts. Slogans such as: "Physicists cannot write good code" and "Computer scientists have no clue about physics" are repeated over and over again. The reason for this is probably the lack of will to understand the arguments of the other side ...

We are not going to describe the software development process methodology here: it is taught in computer science and software management courses, and presented in multitude of books. Instead, let us discuss here the reasons for problems with applicability of it in scientific environment.

At the origin of the problem lays quite fundamental issue of the aim for the computer program. For general-purpose programs, delivering an error-free application, which fulfils the requirements specified by customers are of importance. For commercial products, additional issues of timescales, effort and price are certainly involved. The software development process is applied to optimise and coordinate the effort of programmers to meet these aims. In scientific computer simulation modeling, which follow the requirements of scientific methodology, the aim and the approach are much different: testing and experimenting new models require flexibility rather than stability. Typically it is not possible to specify ultimate set of user requirements and apply software design techniques and tools (such as ER-diagrams, UML modeling) extensively. The need for flexibility is often at the source of the antagonism mentioned above. Software developers, who follow strictly the approaches of software development process are often accused of applying too much constraints and rigidity in the program design, which may limit the possibilities of experimenting with models. On the contrary, physicists who develop simulation codes are accused of "extreme" , "non-professional" or "old-fashioned" style of programming.

Organising and synchronising the efforts of many programmers across geographical and cultural boundaries is an enormous challenge. In the case of large experimental HEP collaborations, various parts of software are contributed and developed quasi-independently, usually in academic environments, where the notion of "deadline" is often not so critical. Integrating such packages, which may evolve with varying dynamics, in order to assemble the complete, functional simulation chain, especially in the fast approaching timescale of the LHC experiments, needs to be achieved. Various approaches are possible, ranging from the situation in which extremely short development cycles and frequent releases establish a pace in which development of reliable and well-tested code becomes impossible, to complete "anarchy" and lack of management. A proper balance needs to be drawn and proper approach still needs to be worked out.

In spite of the fact that the initial, fundamental requirements seem to contradict in many places, certain good practises of the "classical" software development process should certainly be adopted in the software of large HEP collaborations, and the professional experience of software developers, which was worked out thoroughout the last decades, must not be neglected. Among them practises, the most useful seem to be the ones of good programming style (clarity and compactness of the code, good documentation) and clear schemes for versions and releases of the code, the use of code repositories for projects developed as common effort, maintaining backward-compatibility. Fortunately, they are already a part of programming conventions and practices used in the software of the LHC experimental collaborations.

2.9 Elements of computer simulation

The simplest definition of programming is given by the title of the book [35] by N. Wirth , one of the unquestionable experts in computer science:

$$\text{Algorithms} + \text{Data Structures} = \text{Programs} \tag{2.1}$$

For complex scientific computer simulations, this definition, from 1975, is far from being complete at present. I will now expand this list, to include missing elements. The topic of each of the elements is expanded in the subsequent parts of this dissertation.

Data structures

The most important data structure, which in the case of HEP Monte Carlo simulations is the event record, holds the state of simulation. Its proper design becomes yet more important for component-based simulations: the event record plays the role of the interface between the components. It needs to be flexible, to allow for easy modifications of the model. Other data structure involved are the ones used inside the computer models, i.e. to store the state of the model. In Object-Oriented programming, classes, which are the building blocks containing the code, may also be thought of in terms of data structures (or types); in particular, in the component-based simulations, the design of data structures are thus related to the issue of decomposition of the system to components.

In part *III* I will mainly discuss the issues related to event records.

Algorithms

The algorithms encapsulate the functionality of the computer models, and hence are complementary to the data structures, in the sense of 2.1. I would however use this term in a broader contexts of complete schemes that lead from conceptual to computer models. For instance, for the case of HEP Monte Carlo event generators, I would refer to physics process models such as parton shower, radiative corrections or matrix elements, and their computer implementations as algorithms. I will also use this term in its traditional meaning to refer to codes such as (pseudo-) random number generators etc.

In part *IV* I will discuss certain details of the algorithms of `PHOTOS` and `TAUOLA` Monte Carlo generators.

Calibration, testing and validation

In the methodology of scientific computer simulation presented in Section 2.7 the importance of model testing and validation is stressed. This process, which role

is different from the debugging (which is another step in the construction of a computer model), has no counterpart in typical software development. The issues of validation is also discussed in subsection V of the thesis.

Validation and testing of the model may be combined with calibration of tunable parameters and performed repetitively to improve the agreement of the predictions of the model with reference results (i.e. of the measurements). The number of the model parameters may vary significantly. For instance in the case of theories like QED, the only parameter may be the value of the coupling constant (α_{QED}). Conversely, in the detector simulation, every detector channel may be parameterised separately, which may lead to significant number of parameters that require calibration. A good example for calibration is also described in [3], for the new 4-pion hadronic currents parameterisation in the `TAUOLA` generator. The case of `TAUOLA` also indicates that multiple valid calibrations of the model may exist, and the possibility of experimenting with them may give important feedback to theoretical models.

In part VI I will discuss the tests and validations of the `PHOTOS` algorithm. Calibrations and validations are also presented in part II, in the context of the `at2sim` model.

Tools

The development of more and more complex simulations requires that the set of programmer tools is extended beyond simple text editor and compiler. Advanced code building systems, debuggers and documentation generation tools may significantly simplify software development and error detection parts, yet the use of them is not mandatory.

Tools that are, however, important are the ones used for debugging, testing and validation of the models. Source-level debugging (by means of debug statements of the code or dedicated debuggers and code tracing tools) may not be efficient in detection and location of certain errors: the code of the model may be capable of producing results, which seem to be valid, yet are in fact not correct. Higher-level tools, with larger diagnostic power for this kind of errors, such as `MC-TESTER`, described further in chapter 17, may be invaluable.

Another, separate class of tools is related to the conservation and management of the existing simulation codes, and their physical content. I refer here to the `TAUOLA-PHOTOS-F` environment, described in Chapter 16, which makes it possible to prepare versions of the code of the `TAUOLA` and `PHOTOS` generators, suitable to be included directly into the software of various experimental collaborations. This kind of a tool also provides a platform for intermediate developments of the existing codes (e.g. the new parameterizations of the hadronic currents for `TAUOLA`), before the project are translated to modern programming languages, such as C++. The physics content and testing environments built into the original simulation codes usually survive poorly such translation, therefore maintaining the reference implementation is vital for such process. Also, having a tool such as `MC-TESTER` is priceless in such process of translation.

2.10 Validation, trust, and credibility

As already pointed out, the absolute correctness of a model (and hence: also computer simulation) cannot be "proved" by means of methods well known e.g. in mathematics. Instead, the credibility in the model needs to be built in the process of *validation*, which is one of the key elements of the methodology of scientific simulations sketched in Section 2.7 above.

The other important element of the scientific methodology is the requirement of reproducibility of obtained results. For instance, in the case of experimental physics, the observations of new phenomena only become credible after they are reproduced by independent researchers; in theoretical physics - the calculations are checked by peer scientists to eliminate possible errors and discuss the validity of approximations. In the case of computer simulations, which often follow the conceptual models expressed in form of theoretical calculations, new ingredients related to the correctness and *credibility* of the computer code emerge.

Both of the issues: model validation and code credibility, despite being independent problematics, may often be analysed together because of commonalities that can be seen from the point of view of an independent scientist (or: programmer), wishing to use the model as a part of other simulation project.

Let us start with the aspect of the availability of the source code. In contrary to general-purpose computer programs, scientific computer simulations are - practically by definition - open source, and contradict the "black box" approach. The possibility of introducing changes and contributing corrections, enhancements or alternative solutions, which usually requires that the source code is available without restriction, is a natural requirement of the scientific methodology. The availability of the source code also increases the trust to software packages developed by other programmers or researchers and favours code reuse practises. I would like to stress that the sole availability of the source code is not sufficient, even if accompanied by (usually: automatically-generated) technical documentation. Solid documentation of the conceptual model, with clear statements for interpretations, limitations and context of use need to be always provided[2]. Otherwise, the credibility of the results produced by the model may be questioned: the use of a model in a "black-box" manner, without the understanding of its applicability context and limits of validity may lead to wrong predictions (such as non-physical double counting effects in HEP event simulations).

The issue of trust (or lack of it) and code reuse is a more general one, not specific to scientific software. In particular, code reuse practises fail often for complex, unclear, not sufficiently documented codes, and the alternative of "write-it-on-your-own-from-scratch", is chosen. Such choice, the reason of which is typically psychological, despite being more attractive by giving the feeling of creativity, is however not the most optimal one. In the case of scientific simulation the practise of writing own implementation of a computer simulation model "from scratch" is particularly troublesome: the bulk of the work, effort and time needed to create a new model is not in design and implementation, but tests and validation that builds the model credibility.

Another area where the issue of the trust and credibility of the code becomes important is component-based programming, covered in more detail in Chapter 4 . There, the stability of the interface specification and reliability of its implementation becomes critical, in particular for architectures that provide libraries for common low-level services. Due to the requirement of flexibility, which is specific to scientific simulations, the evolution of simulation code (often motivated by i.e. physics models under study) is inevitable. Nevertheless, the changes that involve the common fragments of the code used widely in many places all over the simulation need to be performed with extreme care. For instance, changing the meaning of a elementary function to return $(\pi - \alpha)$ rather than α angle may lead to complete colapse of large software chains used in experimental HEP collaborations - the symptoms of introduced inconsistency may not indicate the cause in all the places where the function is used, and the debugging effort involved in correcting the code may be enormous. Therefore the exact backward-compatibility of the interfaces, including

[2]For some of the codes written in 1970's this remains only a wish...

conventions and specifications used at various steps of the history of the simulation need to be maintained. The fact that the code is freely passed between researchers and various versions of it spread in completely uncontrollable way certainly make the task of code management complicated.

The credibility and validity of the simulation model is usually not visible in the code itself and its value may seem to be in danger of being disregarded. The reality shows, however, this is not the case: the largest part of simulation packages used in High Energy Physics domain, in particular for the LHC experiments, still has a form of FORTRAN codes, the physical validity and credibility of which was worked out as a common effort of many years of work of physicists around the world.

Let us now return to the issue of model validity and credibility, signalled already at the beginning of this section. The credibility of the model needs to be built in the process of model *validation*: continuous confrontations of model prediction with real-world measurements or observations. The fact that a model predictions match with real-world observations for a set of systems does not yet prove its "general correctness". The series of validation tests, involving various contexts and conditions, may only built increased confidence in the model, and usually establish the limits of its applicability (which often corresponds to the limits of the conceptual model, e.g. soft-photon limit).

Validation needs to be performed (and understood) at various level. First of all, the conceptual models needs to be valid, e.g. the approximations used in mathematical formulas needs to be derived correctly, and the validity and applicability ranges of the approximations used need to be established.

Secondly, various errors mentioned in Sect. 2.4.2 need to be eliminated. This requires i.e. that the internal logic of the component code to be valid, and the issues related to the precision of operations (numerical stability, truncation errors) are under control. Wherever possible, the predictions of the computer model should be checked with the predictions of the conceptual model.

Thirdly, in particular for complex simulations composed of components, the validity of the simulation component within certain composition context (and within certain limits of parameters) needs to be tested. At this point, the component is validated within an architecture, as a part of a system. Usually, a component may not be composed in any possible context, but only within a set of contexts that are sensible (and for HEP: are physically correct). The ultimate range of application contexts is, however, often unknown and needs to be probed in validation tests. It is vital to perform both: technical tests (such as event record grammar) and physical correctness tests (e.g. exclude double counting between parton shower and hard matrix element calculations). The results should, as in all validation tests, be confronted with some "baseline" results, being the results of real-world observation (or measurement) or other, credible source of reference results.

Validation of various simulation models contributes an important part of this thesis. I list them here for a reference:

- `PHOTOS`: validation (and improvements) of the simulation model were performed at various stages and contexts:
  - Debuging and various technical improvements of the code: elimination of a bug (induced by rounding-error trap) in the original two-photon algorithm, and kinematic-correcting routine, which made the exponentiated version possible
  - Probing, establishing and extending the applicability context and precision
    - ➢ tests of `PHOTOS` as a high-precision tool in Z and W decays

- ➢ the use of PHOTOS for B,D,K meson decays - universal interference weight significantly raised the precision tag and applicability area

- ❑ at2sim: the validation of the model is described in more detail in Chapter 9; the results of modeling were confronted with the results of measurements performed on various test-bed setups.

Chapter 3

Computer simulations in HEP

The potential of computer simulations was already signalled in Section 2.5. In this chapter I will focus on the use of computer simulation techniques in High Energy Physics. To present the broadness of the scope of use of the simulation in the domain of experimental High Energy Physics, I shall present a "cross-section" through the simulation chain of experimental collaborations in this chapter, and the `at2sim` model of the ATLAS Dataflow System, which will discussed in part *II* of this thesis.

This overview is far from being complete; only the most important area of HEP Monte Carlo simulations in the context of the phenomenology of LHC experiments (in particular: ATLAS) will be presented; in particular, I do not intend to discuss the topics such as the use of simulations in accelerator-related applications, or the ones strictly related to theoretical particle physics (such as Lattice QCD techniques). I do not intend to get into the detailed explanation of detector simulation either. I will provide, however, an overview of the steps and organisation of a full-chain simulation, on the example of ATLAS.

3.1 HEP Simulations as a design and validation tool

The motivation (and ultimately: decision) to build a new high-energy physics experiment come from prediction of physics models, supported by a "proof" of the viability of the, enormous in the case of large, international experimental collaborations, investment of manpower and money. The effort is needed from both: theoretical and experimental communites, therefore a convincing method is to encode the physics models and the conceptual model of detector in a computer simulation, and experiment with such computerized model to explore the discovery potential for the family of tested physics models and ranges of parameters. Initial studies of such kind, that establish requirements for performance of the detector (e.g. spatial and energetic resolution, sustainable rate of interactions) need to be performed before any decision about the design, technology, or geographical placement for the new experiment are taken. Nowadays, one year before the LHC startup, physicists already make plans for the next linear collider, such as ILC [36]. My personal involvement in the studies of the event-generator tools in scope of the ECFA/DESY Linear Collider Workshop [37] was a presentation of the `MC-TESTER` tool during the conference [38].

At the initial stages of design of the new experiment, the goal is typically to probe the space of physics performance, rather that to provide high-precision predictions, therefore flexible and fast tools, such as `ATLFAST` [32, 39] are of utmost value; similarly, universal event generators, such as `PYTHIA` [40, 41, 34], are typ-

ically used, rather than high-precision generators tuned individually to the needs of every experimental collaboration. Of particular importance in such studies are the estimation of cross-sections, rates and signal-to-noise ratios for signatures of physics to be tested/discovered; such information is essential i.e. in the design of the architecture of event filtering and data acquisition systems.

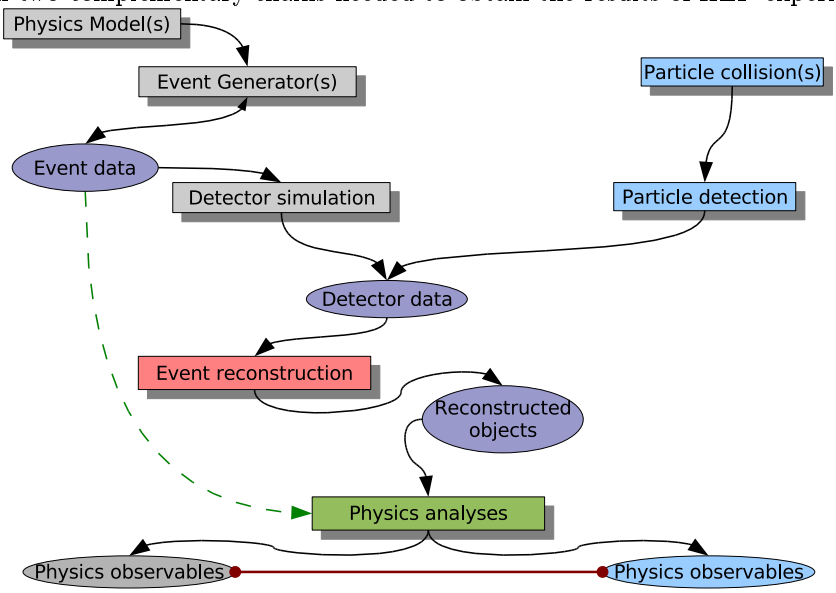
Computer simulations play essential role throughout the whole process of detector design and construction, which may span over decades: detailed detector simulations allow to determine its performance, efficiency without the need of building a prototype or scaled-down model. For the cases where building a full-scale prototype is not possible, simulations are the only method of validation of the design - this is the case for the `at2sim` model, described in the part *II* of this thesis, where the dynamic behaviour of the full-scale ATLAS Dataflow System could only be tested by means of computer model. Computer simulations are the only available method to study the dynamics od full-scale systems based on the behaviour of scaled-down models.

Computer simulations are also necessary in the development of the reconstruction and analysis software. The simulations capable of generating the data in the format that is identical to the one of real detector read-out allow to develop and test the software even before any hardware prototype of a detector is ready. Certainly, the development of the experimental collaboration's on-line and off-line software would not be possible without computer simulations of physics events and detectors.

3.2 HEP Simulations in data analysis

Despite being refereed to as elementary constituents of matter, quarks (and gluons) are not directly observable in the experiments - the state-of-the-art detectors built for the LHC will only be capable of direct detection (and reconstruction of properties) of a handful of types of particles, namely the (already well known) stable ones:electrons, photons, neutrons, protons and a few unstable ones with relatively long lifetime: muons, pions, kaons, even some stable particles, such as neutrinos, or others with energy/momenta below detection threshold will not be detected directly, and only a form of indirect information in form of "missing energy/momenta" will summarise their existence. Already for decades now the discoveries of new particles and high-precision studies of properties of short-living ones are only possible based on the "signatures" of decay processes, which are usually coinciding occurrence of measured decay products, with their kinematics parameters. Typically (a simplification to the case of, say, leptonic decay of the Z boson will be used here) in the process of analysis the mass of the (not directly observed) decaying particle is reconstructed from the masses of (directly measured, or also reconstructed) decay products, and the distributions obtained need to be compared against the distribution that reflects our understanding of all known physics processes that may lead to the signature under study. Resolving the nature and reason for discrepancies between the two is the main task of the experimental analysis process: the unknown parameters of the $theory = experiment$ equation postulated in the first lines of this thesis need to be resolved. Due to the more and more complicated nature of physics processes under study, and the non-ideal nature of the detector that are used (limited positional/energetic resolution, unstable (from run to run) geometry and magnetic field, ageing of electronics due to radiation), the only way to take into account all the ingredients is to use complex computer simulation, often referred to as "full chain" simulation, the elements of which will shortly be presented below. The physics analysis does not attempt to completely disentangle the convolution of all the physics processes that happen in the event and in the detector. Instead,

Figure 3.1: Simulations (grey boxes) and experimental data analysis (blue boxes) form two complementary chains needed to obtain the results of HEP experiment.



it tries to understand and parameterise the response of the detector, combine it with the parameterised theoretical model expressed by means of Monte Carlo event generator, and perform the comparison of theory and experiment at the level of distributions. Fitting of the parameters of the theoretical model so that the distribution produced by simulation and extracted from experiment agree best is the indirect way of "measuring" the parameters of the theoretical model. For the cases where objects such as quarks are the subject (or part) of the analysis, and the QCD need to be confronted with experimentally detected hadronic jets, non-perturbative models for the processes of hadronization and partons shower need to be employed, and multi-stage computer simulations (involving a variety of models: from hard-process parton-level calculations, down to detector response and signal shaping) are the only possible way to perform the, "comparative", studies.

The diagram in Figure 3.1 presents the flow of data and activities that need to be involved in order to obtain physics results from a high-energy physics nowadays. Two branches: experimental (blue boxes, on the right) and theoretical, expressed by means of simulations (violet boxes, on the left) produce the results, which are subject of physics analyses; the results of the analyses performed on both branches need to be confronted with each other - the agreement signifies that the theoretical model(s), expressed by means of the simulations, describes the phenomena observed in the experiment, and hence builds up the confidence in the theoretical model. In the case of significant discrepancies between the results obtained from experimental and simulation data, the model would need to be improved, for instance by tuning its parameters or using other approximations; the discrepancy may, however, also indicate a problem with the experimental setup or insufficient understanding of the detector response. In all the cases investigations need to be started - they are usually performed on the branch leading through the computer simulations.

The computer simulation play another essential role at the time of data-taking and reconstruction: having a functional model of the detector allows for diagnostics of the detector problem (testing hypotheses of problem causes), allows to better understand the performance of all parts of the detector and makes it possible to im-

prove (or: optimize) the reconstruction algorithms, as the process of reconstruction and the detector performance are understood better and better.

3.2.1 Full chain simulation

In the HEP data analysis process a large computer simulation, also called "full chain" simulation needs to be executed. It contains a sequence of steps which start from theoretical prediction embedded in event simulation, followed by detector-response simulation, results of which are fed to the reconstruction software and analysis. (This way also the reconstruction software and analysis code themselves may be studied! see 3.1). Let me now briefly present these steps, which are often performed independently by means of dedicated programs, on the example of the ATLAS experiment; a thorough overview of the simulation process is also provided in [42].

Event generation

The first step in the chain is the simulation of physical events by means of HEP Monte Carlo event generators. Unless for the case of design studies, where, the discovery potential and overall performance of the new detector is studied, or new models are tested, the events used in the full chain simulations are, so called, "minimum bias" events. A strict definition of a "minimum bias" cannot be provided; in particular, experimentalist and theorists usually maintain their own, differing views. In the simulation of minimum bias events it is not sufficient to model solely the "signal" process, which for the case of proton-proton interactions at LHC energies is often studied at partonic level. Instead, the results of particle collision needs to be modeled to reflect the real collision in as much detail as possible. This involves that the effects of multiple interactions, beam remnants and elastic scattering processes are properly simulated. For hadronic interactions, non-perturbative processes of parton showering and hadronization needs to be fully treated. Cascade decays of short-living particles and resonances need to be evolved. The effects of QED bremsstrahlung may also be taken into account at this stage[1].

The simulation of the event is usually performed by a set of collaborating simulation programs, dedicated to certain tasks, and exchanging the data through the event record. For instance, the "signal" or "hard" process may be generated by a dedicated hard-process generator, parton shower and hadronization can be performed using the generators such as `PYTHIA` or `HERWIG`, the decays of τ leptons may be generated using `TAUOLA`, and ultimately bremsstrahlung photons, being the effects of QED radiative corrections in particle decays may be generated using `PHOTOS`.

Minimum-bias events are often discussed in a more general scope of the underlying event model. These non-perturbative, parameterised models play important role in the data analysis. Proper choice of parameterisation and calibration of the underlying event model is a delicate task, which may affect the precision of obtained experimental result.

The way in which the event is usually simulated (or: "generated"), i.e. as a multi-stage process, in which a multitude of simulation codes participate, determine the form of the output of this simulation stage: a list of objects stored in the event record, with their properties, relations and simulation-specific event history information. In the first approach, it would be sufficient to select the final-state, non-decayed or stable particles from the event record and transfer them as an input to the next step in the simulation chain, namely detector simulation. This is,

[1] the effects of QED radiative corrections are also sometimes treated together with detector efficiency simulation, or the experimental data may be treated with an estimated impact of their result to form "QED subtracted data".

however, often not sufficient for the sake of studies of data analysis code. To be able to answer questions, often unphysical yet possible to answer at the technical level, such as "what is the flavour of the quark associated to a jet", the availability of the complete event generation history (and hence: complete event record) would need to be stored. Additionally, when the final-state particles from event generation steps are used to seed the list of "primaries" in detector simulation step, their logical association needs to be stored in form of so called "*MC Truth*" information.

In the case of the ATLAS off-line software, the output of the generation step is stored in Event Collections, in the `POOL` permanent object storage area, in the form of `ROOT` files. The Event Collections resemble the n-tuples and contain snapshots of the event record (in the `HepMC` format), accompanied by a container with *MC Truth* information.

Detector simulation

In the second step of the full simulation chain, the response of the detector to the events produced in the previous step is simulated. Due to the complexity of the simulation methodology and problematics, I do not intend to present this topic in detail here.

Detector simulation stage allows to accurately model the efficiency of the detector, taking into account factors such as detailed geometry and alignment of detector parts, magnetic field map, existence of non-active constructing material, degraded performance of detector channels caused by radiation, etc. Taking into account the impact of all these effects, with sufficient precision, in any theoretical calculation is practically impossible, and the computer simulation of theoretical model of an event, followed by detector simulation, remains the only way of obtaining high-precision predictions for measurement results. The possibility of performing this type of simulations, and hence exploring and understanding the detector, is vital not only at the detector design or data analysis step, but also to diagnose the detector problems when they arise.

Detailed detector simulations are built, improved and maintained throughout the whole lifetime of the detector. Nowadays the model is usually build using the dedicated detector simulation framework: `GEANT` [43]. The simulation requires a list of primary particles (with 4-momenta and position) to be specified as input data. Because the majority of the interaction simulated in event generation step occur at microscopic distances, it is usually sufficient to use the list of undecayed/stable particles from the event generation step as an initial list of particles for detector simulation. As the simulation of the event evolves, the track of each particle in the event is simulated, according to the direction of its flight. As a result of interaction of particles with the material of the detector, secondary particles may be produced, and the energy/momenta decreases, up to a point where the particle is "stopped" in the detecting material. Secondary physical processes related to interactions of radiation and high-energy particles with matter, such as conversion, bremsstrahlung, transition radiation, or scattering are also simulated.

The simulated model of detector contains geometrical volumes defined as active. If a simulated track of a particle passes such active region (and also some additional requirements are met, such as the particle energy being over a certain threshold, etc), a "hit" is registered: the model "detects" a particle. The information about the geographical coordinates of the hit (e.g. the identification of the active "cell" that was hit) may also be accompanied by more details, such as energy threshold, energy deposit or the characteristics of analog signal.

The initial result of the detector simulation: lists of hits are usually further re-processed to simulate the detector read-out system. This stage of simulation, which may also involve the simulation of event pile-up, is often a separate step of

the full chain simulation. Once the digitisation step is finished the format of the output (GEANT "digi" data are usually converted to the "byte-stream" format) from simulation is the same as the data read out from a real detector, hence it may be fed directly into subsequent steps of analysis, namely event reconstruction and analysis.

The full simulation of propagation of particles coming from a collision through the detailed model of detector is a resource-consuming task. In the case of ATLAS, full simulation of a simple $Z \rightarrow ee$ event may requite ~15 minutes of CPU time per event! It is therefore hard to apply for various physics studies, which need to perform simulations on various samples of events with long statistics. In such analyses, in particular in the ones that need to give approximate estimates of signal and background for specific channels of interest, the high-precision of the detector simulation is usually not a priority, and fast, parameterised simulations of detector may be used to perform initial studies with significantly less effort. ATLFAST [32] is the particle-level fast simulation package that was used widely in ATLAS. The ATLFAST package was initially written as a stand-alone FORTRAN code (and also ported later to C++ with ROOT infrastructure - I participated in this project)[39], and later it was adopted to the ATHENA framework [44], using GEANT package. The simplified parameterisations used to model the processes and objects in ATLFAST were exact enough to allow to explore the discovery potential for a variety of physics processes expected in ATLAS, and documented in the ATLAS TDR document[45].

Reconstruction

The digitised response of the detector (both: simulated or real), in form of raw byte-stream is treated in the process of event reconstruction, to identify physical objects, such as hadronic or electromagnetic jets, isolated particles and missing energy/momentum. At this stage a variety of dedicated reconstruction algorithm, specialised in a certain part of reconstruction is used, e.g. track finding and extrapolation, calorimeter cluster building, muon reconstruction, etc. As a result of (often: multi-stage) process of reconstruction, objects that have physical meaning, such as "electron", "jet", or "particle track", which in turn can be used in physics analyses, are created.

In ATLAS, the reconstruction produces two types of data: the Event Summary Data (ESD), and Analysis Object Data (AOD). Firstly, the digitised (bytestream) data is converted to ESD, which contains the complete information extracted by reconstruction algorithms. ESDs are usually not directly usable for physics analysis, therefore they are further distilled to AOD, which organises the data into analysis objects with physical meaning and may be examined using analysis algorithms. An essential aspect of the AOD creation for full-chain simulations is Monte Carlo Truth association, which directly links the original Monte Carlo event data to the reconstructed objects, enabling the user to assess the performance of the algorithms.

Physics analysis

The final step of the full chain simulation is the physical analysis of the reconstructed data. The analyses, written by physicist, employ statistical analysis techniques applied to the features extracted from reconstructed objects. Histogramming and fitting tools are widely used in the analyses, which may also be performed interactively.

3.3 Physics embedded in computer codes

As described in previous section, computer simulations are essential "carrier" element allowing to confront, often abstract, models of theoretical physics with phe-

Figure 3.2: Examples of detector simulations, taken from [46]

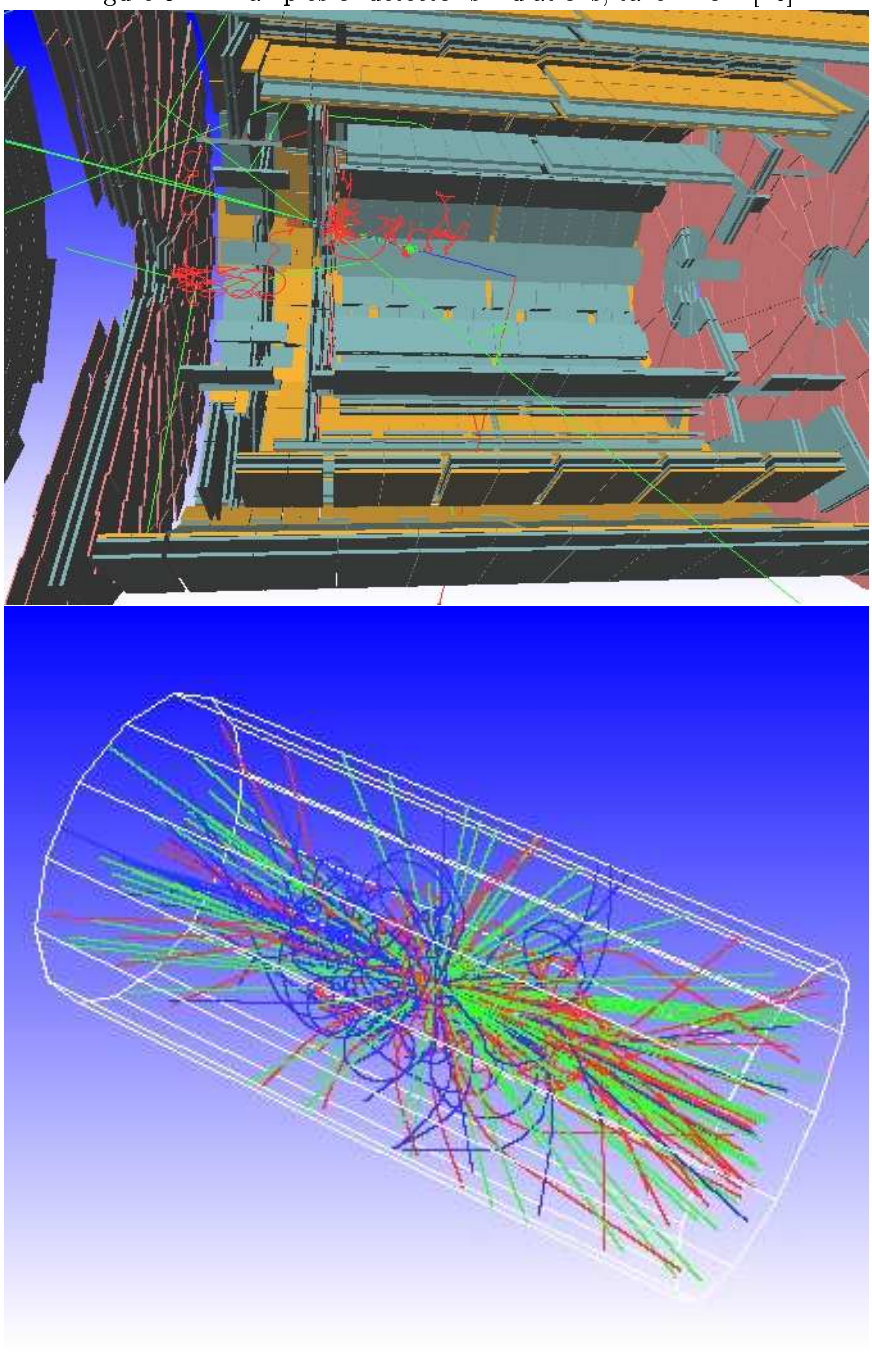
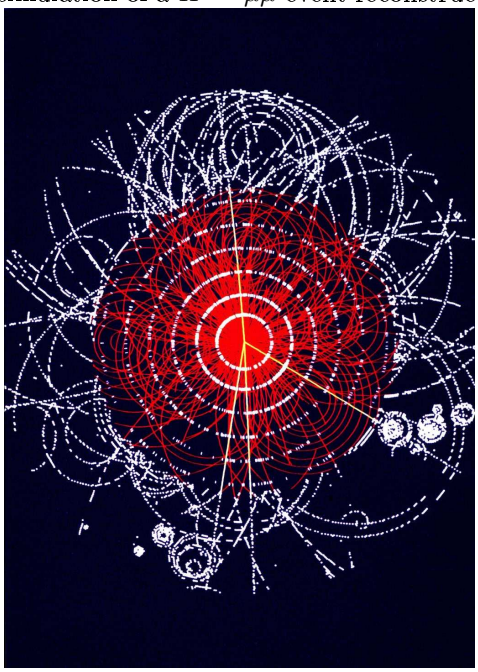


Figure 3.3: A simulation of a $H \rightarrow \mu\mu$ event reconstructed in ATLAS



nomenology of experimental results. Monte Carlo event generators are therefore the most convenient (and most frequently chosen) way to express the theoretical models for High Energy Physics. From the point of view of the experimental data analysis, dedicated semi-phenomenological models are often used to allow for best fit to the experimental data; to express such model without the context of the experimental data and experimental setup would often be impossible or at least unnatural and difficult, and computer simulation remains the only possibility to conserve the model (i.e. for future use in context of another experiment, comparison of results, etc). These computer simulation codes may indeed contain very rich physics content, that spans far beyond theoretical models expressed by analytical formulae and the values for the model parameters that need to be determined based on experimental data.

The fact that the simulations play the role of the mediator between theoretical and experimental domains of high energy physics determines the specifics of evolution and "internal life" of their codes inside and between experimental collaborations. I will use the example of `TAUOLA` for the discussion here. Each experimental collaboration (i.e. in a group of collaborations performing similar kind of studies) usually maintains its own version of Monte Carlo event generators, which, with time, differentiate from the "master" version, as provided by the generator author (who is usually a theorist, and encodes some theoretical model in the generator); new parameterisations for models are introduced and original physics initialisations are altered to obtain best fit with data analysed by the collaboration. For the case of `TAUOLA`, at least two such branches (in addition to the "master" version, as published in [28]) were identified by the authors, namely the ones used by CLEO and ALEPH experimental collaborations (see also Chapter 16 for discussion of `TAUOLA` version management). The codes used in the analysis software of experimental collaborations not only contain the tuning in form of initialisation constants, but also genuine algorithms, or fragments of code that differ from official distribution version of generators. It is important not only to preserve the code for future re-use, but also to maintain the contacts with collaborations, to realize (and "track down") the

spread of various versions of the code. Matching these requirements of maintaining various, often old-dated versions of code with the need for development of the new features become non trivial task not only from technical point of view.

Chapter 4

Component-based computer simulation

In this chapter I will discuss the use of components in computer simulations. The use of components seems to be one of the leading subjects of discussions nowadays for discrete-event scientific (andmilitary [47, 48]) simulations. The area of HEP Monte Carlo simulations seems not to be present in these discussions. Hence, I will firstly present the topic as described in the available literature and publications, then I will present our own observations and conclusions for the use of components in the HEP Monte Carlo simulations.

One of the most often referenced, historical paper is the one by D.L. Parnas [49], dated in 1971. Already there, in the Abstract, it is stated that

> "(...) the effectivness of a modularization is dependent upon the criteria used in dividing the system into modules(...)". A "conventional" and "unconventional" decomposition of a system is prepared, and the important conclusion is drawn: "(...) it is almost always incorrect to begin with the decomposition of a system into modules on the basis of a flowchart. I propose instead that one begins with a list of difficult design decisions which are likely to change. Each module is then designed to hide such a decision from the others. Since, in most cases, design decisions transcend time of execution, modules will not correspond to steps in the processing.(...)"

4.1 Component-based approach

In this section I will present the concepts and terminology of component-based software development, and its applicability to computer simulation. I will rely on the description given in [50], which discusses the component approach in context of discrete event simulations. My personal experience with component approach for (discrete event) simulations is based on its application in the `at2sim` model (Chapter 9). I believe this experience may be used in the area of HEP Monte Carlo event generators, where the use of component-based approach is in very early stage. Apart from the GAUDI architecture[51], the use of components does not seem to be present in HEP simulations, and no general discussion of this topic can be found in the currently available literature, whereas for the discrete-event simulation (used also for military application) the issues of definition and reuse of the components are the areas of vivid discussions nowadays [47, 48, 50, 52]. The component approach

to design of the computer simulation is still not as widespread as in the branches of commercial or industrial software.

Modular approach of HEP MC simulations

Growing size and complexity of simulations, initially written as monolithic programs, turned the attention of the modellers to the issues of the code reuse and hierarchical organisation of the simulations. In 1970s, the modular approach to simulation became widespread. Motivated by the availability of modular programming languages, such as FORTRAN, the simulations were written as functional blocks, which could be programmed, compiled, debugged and tested separately, then linked together to form the main simulation program. The benefits of this approach became apparent, and there was a lot of interest in the organisation of the modular software architectures and use of components. Also in the field of HEP Monte Carlo simulation, the trend for improved modularisation was present, especially in early 1990s, in scope of projects such as `COSMOS` or `ELOISATRON` [53, 54]. These efforts, unfortunately, did not bring a major breakthrough (change from modular to component-based approach), yet they consolidated the efforts in defining the common standard for the `HEPEVT` event record. Mainly due to FORTRAN-based architectures of collaborations' software chains, and the need for maintaining backward-compatibility, the majority of the new simulations in 1990s were written in modular FORTRAN[1].

Motivation for component approach

The "classical" architecture for HEP simulation with simulation modules exchanging the data by means of central event record data structure proved to be effective for more than a decade. A question of reason for changing this approach, to the one based on components, arrives in a natural way.

The key arguments of the component-based approach are increased reusability and exchangeability of the elements (components). The modular, event-record based approach of HEP MC simulation already provides exchangeability and reusability at the level of modules. The component-based approach allows to employ the reusability on the much finer (or lower) level, and introduce an "architecture", which (even though can be thought of as "constraint") allows for better organisation of the modules' code, which becomes important as the simulations become more and more complex. My aim is therefore not to discredit the event record approach (which is thoroughly discussed in part *III* of this thesis), but rather to postulate that component-based approach should be explored in the field of HEP MC eveint simulations, because of the profits and widespread use (including HEP experimental collaboration software chains).

4.1.1 Components and Architectures

The idea of component-based software engineering is concentrated around two central terms: *components* and *architectures*. In the process of *composition*, the components are linked together according to rules and mechanisms defined by architecture, to form the final application (or: the model of the system in computer simulations). Because of popularity of this approach there are many definitions for these central terms being currently in use.

[1] Efforts to implement "Pseudo-class" modules, where data and functions (methods) are bundled together were, however, successfully implemented in FORTRAN codes, i.e. in the KKMC generator [55].

A component is usually refereed to as a unit of composition used in software development. Such definition would not, however, bring any conceptual difference between modules and components. Therefore, the component needs to fulfil additional requirements:

- it needs to have a specified, clear and well-defined *interface* (public declaration of its functionality), with good documentation;
- the interaction with other components (i.e. not being the sub-components of itself) may only be realised by means of the interface.
- it should be possible to deploy it independently, as a subject to composition by third parties; this may (but does not have to) mean: application independence
- it usually introduces a level of abstraction,

The most important of these requirement, which in fact differentiates the component from the modules, is the one concerning the interface.

The second of the terms, the architecture, is usually understood as an organisational structure of the system, which is built from entities such as subsystems, components or classes. The architecture defines the actual interactions and relationships between its parts.

The architecture often defines also the principles and guidelines that governing the design and the evolution of the parts, hence, it is often not possible to speak about components without the architectures. This also means that usually the components are developed with certain architecture in mind. This is motivated by the practical reasons: typically, the components that needs to be developed are not very generic (such case being an "ideal" component). They should rather be deployed within certain context, which may either be dictated by common sense and applicability, or restricted due to, constraints originating from e.g.: physics laws or the approximations used in the conceptual model of the component.

Please note that in the event-record based approach, the use of commonly agreed standard for the event record, `HEPEVT`, plays the role of the interface specification; the difference is that event record specification is limited to the functionality related to data-exchange between modules, and does not specify how to declare, or "call" functionality of other modules.

4.1.2 Reusability, extensibility and evolvability

The most attracting promise of the component approach is reusability, which may be involved at various levels; first of all, as the reusability of the complete components, that may be deployed in a "black-box" approach, in an easy way. Secondly, the libraries of components should be designed with reusability in mind; in particular, Object-Oriented design approach may make the code of the components easier to understand (e.g. by introducing levels of abstraction), and more flexible: the component may be "tailored" or "extended" to specific context of use by means of e.g. inheritance.

In the case of the `at2sim` simulation I explored this multi-stage context of reusability in the following way: the components were written for the specified architecture[2], the `at2sim` architecture, based on the architecture of the Ptolemy framework. The components (models of the nodes) could easily be composed to create simulations of various systems: ranging from small "testbed" setups used to

[2]The model of the Ethernet Switch being an important case for component reuse here: it could be deployed also in the architecture of the `OPNET` package.

validate the model, to the large simulations of various architectures of the final system. Because of Object-Oriented design of the component library, it was easy to prepare specialised components, thus add and integrate the new models.

Let me now turn to the more complicated case of the component being re-used outside of its "native" architecture. Apart from the technical issues, which are described in next subsection, there are also other potential pitfalls in the use of component libraries. The most important of them arises from the availability and complexity of component libraries. The already large number of various component libraries makes it difficult to determine the applicability of certain one, and the amount of work needed to evaluate (which requires: to learn) "yet another" architecture and library and overcome the technical issues with integrating it into own architecture may be discouraged. Secondly, the component libraries nowadays are often large, and often lack sufficient documentation: the availability of template libraries in C++ and automated documentation generators result in component libraries containing hundreds of classes and functions, with no documentation of which ones are the important ones, and which one are used rarely or limited to internal use. Such libraries also try to solve the problems of low-level functionality on their own: the technical details of object persistency, input/output, etc, which in general should be addressed by other, dedicated components, "bloat" the libraries with the complexity. From the technical point of view, such "all-in-one" approach (providing the whole required low-level functionality) usually leads to better performance. It, however, also often imposes that its architecture is accepted as the "main" architecture of the system. Such approach is often not acceptable and leads to the decision of... writing own component library, adapted to own architecture, rather than using an existing one. Certainly, the issue of lack of trust in other's code (see Subsect 2.10) is an important factor motivating the choice of component library.

The decision of writing an own component library rather than re-use existing one may, however, turn out to be very risky, in particular for scientific code, because of the issue of validity and tests. The modules with long history, such as `PHOTOS`, `TAUOLA` or `PYTHIA`, have credibility built by thousands of hours of CPU execution time, hundreds of various applications and - what is most important - crosschecks with experimental results. Therefore, despite the fact of being written in modular FORTRAN rather than modern object-orientated language such as C++, they are (and will be, at least for initial stage of the LHC data analysis) favoured by experimental collaborations, which prefer them rather than the new codes, with not-so-strong and long-lasting credibility evidence. Hence, at least for the case of the LHC experimental collaboration, the favoured method of building the (usually C++ - based) software was to build wrappers to existing codes, to enable their use in the new architectures. An alternative approach, the complete rewrite of the existing code, combined with complete change of the internal module's architecture, is also being tried out (`HERWIG++`, `GEANT4`), however rebuilding the credibility to the code requires significant time and effort. The third, intermediate approach, i.e. the rewrite of existing code to, say, C++ with as small changes in the internal logics as possible, also seems to be viable (`Photos+`[56], plans for "*TAUOLA++*"), yet such effort seems not to be justified - the experimental collaboration did not express strong needs that would motivate such translations.

Let us now shortly discuss the issues of evolvability and extensibility. These two promises of the component-based approach are vital to HEP simulations. Following [50, 52], an extensible system is one to which new components may be added, and these new components are capable of interacting with others, already existing in a system. Evolvability, in turn, means that new components may be used (composed) in place of older components, and this way provide improved quality (or: precision), while providing the functionality of the original ones. In HEP simulations both

features are important: not only the individual models may be improved to provide, for instance, better precision, but also new (conceptual) models may be proposed and need to be tested. The need for evolution and extensibility are in fact "native" features of scientific software, and therefore differentiate it from other types, not requiring it to such high degree. In contrast to typical commercial applications, the need for extensibility and evolvability is not to fulfil the requirements of users (which, despite evolving, are often predictable and - in fact - often "static" in their nature), but rather to constantly change and test new models, following the rules of scientific method.

Together with the two abovementioned features the issue of dynamic binding is often discussed. It means that new components may be incorporated into the model at run-time, i.e. they may be developed much later than the model itself, then seamlessly integrated with running model, without any need for recompilation or even stopping of the model. Such techniques are becoming more and more popular in commodity computer programs now by means of "plugins" or "extensions" (e.g. a "plug-in" that allows to view a PDF file in a web-browser). In the context of HEP simulations, at least at the current stage, the need for such mechanism is not strong. Despite that, certain packages, such as `ROOT`, provide the possibility of loading/executing portions of compiled or interpreted code on demand, at run-time. I used this feature heavily in the `MC-TESTER`–based studies of `PHOTOS`, described in Chapter 18, and I found them to be helpful. One should also be aware of the technical limitations for dynamic binding due to the technologies used in various programming languages: certain languages, such as Java, deal with the problem of late or dynamic binding gracefully and natively. Others, such as e.g. C++ may need careful treatment, in particular in the cases where multiple inheritance is needed and the code of base classes is changed - a recompilation/linking step may be required then.

4.1.3 Adaptors, Wrappers, Translators

The most important feature that defines a component is the presence of the interface, which is a specification of possible interactions between the components. When used in a "native" architecture, all the components usually present a consistent interface, use the same conventions, suggest the natural context of use. On the contrary, when used in other architectures, they need to be adapted to the architecture, in which they are supposed to be used.

Following the classification in [50] one could identify three methods for matching the components to architectures - by means of wrappers, translators and adaptors. Let us shortly describe their applicability here.

Wrappers are constructs which are most widely used to solve the problem of code reuse. By definition, these are container objects, which does not have any functionality added, but only implements the set of interfaces specified by an architecture. Wrappers are used to adapt a module or a component written in other programming language, to a specific architecture: a "wrapped" module becomes a component. A wrapper, like a component, only makes sense within a specified architecture.

Adaptors are components[3] that are used to adapt the components designed for other architectures. The adaptor changes the way in which a component presents itself to other components. Adaptors are used to reduce the complexity of interaction, and they may be organised as "adaptor layers". More than one component may use the same adaptor.

Translators (or mediators) are used to adapt the "protocol" used in interactions

[3]unlike wrappers which are only "container" objects

between the components. It is usually used in context of applications that makes use of "events" to implement the interactions - the translator will accept event types produced by one component, and translate them to corresponding events "understood" by another component[4].

In the simulation chains used by the LHC experimental collaborations, the wrappers are amongst the most widely used components: they usually interface the event generator codes written in FORTRAN to the collaboration software architectures, such as `ATHENA` [57] (which implements `GAUDI` architecture [51]) .

4.1.4 Components and Object-Oriented programming

In the Object-Oriented programming the use of components is a natural consequence of programming paradigm, language constructs and ... common sense. Let me now shortly summarise the features of OO-programming that are relevant to component-based approach: these are practically all concepts of OO-programming!

Object Orientation assumes that the code of the program is composed of modules, called objects, which encapsulate data and the code (methods). Typically the objects are hierarchical, compound constructs, and interact with other objects by message-passing, i.e. one object requesting a service from another. Objects are created in programs as instances of classes. In the analogy to variables and data types, classes - as it is for data types - define the overall behaviour and application contexts (e.g. logical variable may hold two values, a set of (Boolean) operators is defined), and objects - as it is for variables - are instances of classes, holding the actual data, on which methods (such as operators) are used. The code that refers to object-related operations (methods) is therefore written in context of classes, whereas objects (instances) are actual entities in the program.

The mechanism of inheritance allows for extensibility: a new class may easily be created as a specialised sub-class of another class, while polymorphism and virtual methods allow to use the instances of sub-classes in the contexts where objects of super-class are expected. This way, duplication of unchanged parts of code may be avoided.

The definition of the class may also be treated as public interface specification: a list of publicly accessible variables and methods of a component.

Despite the fact that OO programming languages seem to be so well suited for component-based programming, they alone can not guarantee extensive reusability of the code. The reason lays in the variety of possibilities of system decomposition into classes (which in the case of HEP simulations may be affected by requirements of the laws of physics). Neither of the two opposite approaches: a single, complex, solid class or a set of many simpler classes seem to be optimal here; the former usually provides a more compact code, yet the component which contains the whole functionality in a single class may become far too complex when it comes to modifying (which requires: understanding) of its code. In the latter, where the component is implemented in a set of classes, often assumptions and constraints are introduced to implement internal architecture organisation, which in turn may cause conflicts with the architecture into which the whole component is integrated. Such "nested" component approach is natural and should not be avoided, yet one should consider the implications of maintaining separate micro-architectures and infrastructures for each component.

[4]In at2sim, the NIC model acts as a translator: it takes Amessage objects, which represent high-level Data Collection messages, and converts them to *GigEPacket* objects, used by models of the Ethernet switch. Not only the "packetization" of the message is simulated, but also the Amessage-typed event is translated to *GigEPacket*-typed event.

4.1.5 Examples of use of components

The use of components in software development is widespread nowadays. Let us give examples of successful use of component techniques, mainly in Open Source projects. The use of components in commercial software written for Windows operating system is even more widespread; and Java-based products use them extensively almost everywhere.

Components are mostly noticeable in the user interface environments: interactive graphical objects such as buttons, sliders, menus are common elements of almost every program with graphical user interface. In fact, they are often refereed to in books describing OO programming and components approach. Component approach to the GUI (Graphical User Interface) is present from the very beginning in X11: the graphical systems used in UNIX/POSIX operating systems. The design of the X11 windowing system made it possible to implement multitude of libraries of graphical components, with notable example of MOTIF library, most popular in early 1990's.

Increasing popularity of the GNU/Linux operating system and the Open Source movement, new "toolkits" of graphical objects became available, such as TCL/Tk, FLTK, etc. The most notable nowadays are GTK and Qt, which form the basement for much larger projects: GNOME and KDE, desktop managers and application frameworks. Both projects are state-of-the-art examples of the use of components approach. Let us focus on one of them, namely KDE/Qt. Qt[58] is a C++ portable (available on UNIX/X11, MacOS and Windows platform) library of components, providing a framework for graphical-user-interface (GUI) based applications, developped by Trolltech company and available on commercial and open source licenses. It is not only an extensible library of GUI components, but also other general-purpose components that deal with thread handling, XML, database access, network communication, etc. The components provided by Qt are grouped in separate, quasi-independent modules, dedicated to certain tasks, in order to minimise the overhead and maintain clarity and good performance. The overhead of the required internal architecture (infrastructure) is kept to a minimal size. Qt also provides extended message-passing mechanisms, not available in standard C++, namely slots, signals and properties.

Based on Qt is the KDE graphical environment: a desktop manager and rich set of applications based on common, framework and making use of component approach to a high degree. KDE is also an architecture for the new applications, or extensions of existing ones: the applications in KDE are built from KParts, which are high-level, pluggable components, which may be easily re-used in other application. An example of KPart may be a viewer for a PDF or PostScript file, a window with HTML browsing capabilities, a text editor, a spreadsheet, etc. The most powerful demonstration of the KParts technology is konqueror: a universal web and file browser, which is capable of displaying any KDE-supported file inside its window: it loads a proper KPart component, depending on the type of the file. Another example is IOSlaves mechanism, which provide an architecture for implementing file-related operations. For instance, the http (web) protocol IOSlave may be used in a web-browser in order to get the file from a web server, but other IOSlave may be used as well, e.g. ftp, ssh-based, webdav, CVS, etc. The state-of-the-art components were noticed by commercial partners: KParts responsible for displaying HTML and JavaScript (web-content) in i.e. konqueror web browser were adopted by commercial vendors such as Apple and Nokia, who further improve them (as WebCore/JSCore) and use in their commercial products (i.e. Apple's succesful Safari web browser).

4.1.6 Component-based architectures in HEP simulations

The traditional, LEP-era approach to the modular organisation of the large HEP simulation with central (HEPEVT-based) event record became insufficient for the LHC experimental collaborations, which decided to use Object-Oriented approach for their software development. This decision, motivated by the complexity of the new experiments evolved (in a natural way) to the use of component-based approach and architecture-based design of the collaboration software. The availability of the GAUDI architecture [51, 59] specification and ROOT framework[60]) allowed to create component-based architectures for the off-line software of the four LHC experiments:

- ATLAS implemented ATHENA framework [61, 57, 62], based on GAUDI architecture
- LHCb implemented GAUDI framework[63], based on GAUDI architecture
- ALICE implemented AliRoot[64], based on ROOT (see also subsect. 11.6.2)
- CMS is in phase of migration to EDM (Event Data Model) architecture[65], which is based on experience from BaBar, CLEO, CDF, D0, and previous CMS system; EDM is custom component-based architecture based on central "Event", it is focused on Grid technologies and produces browsable ROOT trees)

A common, point of difference in all of these architectures, is the access to event generator data, which is solved differently by each of the collaboration and requires interfacing to FORTRAN event generator codes and/or HEPEVT event record. The effort of writing event generator interfacing and event-data exchange routines needed to be performed repeatedly by each of the experiments. Lack of common standard for event record data structure or architecture is apparently visible; the biggest promise of component-based approach, reusability and exchangeability of components, was not possible to realize in this context... The work related to the development, debugging and testing of the interfaces to the event generators seems to be one of the most important efforts in the domain of event-generation domain of collaborations' simulation chains - the stable versions of code of event generators are already provided by their authors, and even managed in scope of the GENSER project [66][5]. This activity requires not only fluency in the use of the architecture and writing own components and writing C++ - FORTRAN bridges, but also excellent knowledge of the physics module being interfaces.

In the area of HEP Monte Carlo generators, the component-based approach is still in its early stages: the existing projects of C++ generators based on ThePEG architecture (see also subsect.11.6.8) or SHERPA [67], despite evolving, have not been able to provide a high-quality, reliable and complete generators, that could be used in the collaborations' software chains. Incompatibility of these architectures (between each other, and in relation to the architectures used by LHC collaboration software chains), and the use of ad-hoc solutions for transferring the data to other generators (via HEPEVT, HepMC event records, or data files) does not promise easy integration in the future either.

Let me now raise the problematics of requirements for the architecture for HEP simulations. Contrary to the "standard" software architectures, which, once establish, does not need to change, one of the main requirements for an architecture of HEP simulations is flexibility. Due to the evolution of the experiment, need of use of new theoretical models encoded in new event generators and various optimisations

[5]The issue of maintaining stable versions of external modules, such as event generators, is essential for the consistency of the collaboration software, yet it does not fall into the category of components and architectures. In the past the role of central CERN repository for physics codes was played by CERNLIB [26].

of the reconstruction and analysis stages, new requirements, often contradictory to the ones specified (and assumed) initially need to be fulfilled. Already for LEP experiments, the "decay" of, initially highly-organised, software architectures, with time, has been observed [68]. For the LHC collaborations' software, time will ultimately verify the flexibility-versus-elegance design decisions for their architectures.

Definition of a flexible architecture for HEP simulation, as opposed to "standard" software architectures, cannot be performed solely on the base of a set of stable requirements expressed by a majority of the users: it often requires skills related to sociology, and understanding of relations and dependencies existing across, so differing, points of view of theorists, experimental physicists, and ... software engineering experts. Gathering consistent and complete "user requirements" in such situation is practically impossible - the requirements evolve constantly[6] (and may even happen to be contradictory). Taking into account only the requirements of one group (e.g. only the theorists, authors of event generators) and neglecting the others will certainly effect in lack of flexibility of the architecture. Any assumption on the model of the data, and interactions (such as assumption of tree structure of the event record) would also result in the architecture appearing to be too rigid for some applications. Imposing such limited architecture "by force" as a standard would not be successfully either: a redesign (or adoption of another standard) will happen anyway as soon as the first critical shortages start to appear.

Ultimately, let me point out one more issue related to the definition and use of architectures: the definition (and enforcement) of the low-level infrastructure. Having in mind usability and potential simplification of the components' code, some architectures provide the automatic integration of low-level mechanisms, such as persistency (`ROOT`, `ThePEG`), or "blackboard model" for transient data store (`ATHENA`). Such approach may indeed simplify the development of physics codes (e.g. the developers with less "technical" point of view, does not need to care about these aspects), yet it also introduces an important threat to the project: in case that the mechanism (or whole approach, such as "blackboard model") does not fulfil expectations, and becomes a limitation, significant parts of (the physics components) code may need to be completely redesigned and rewritten, for the new architecture. Moreover, assumptions of specific mechanisms being provided by architecture greatly complicates the possibility of the component being interfaced to another architecture (e.g. reuse of a part of physics analysis written for ATLAS would be relatively easy in LHCb, at least from architectural point of view, whereas for CMS or ALICE would probably not be possible at all.

4.1.7 HEP components definition criteria

The first stage in the process of computer simulation,as described in subsect. 2.7, consists of defining the performance, precision, and extent of modelled functionality for the simulation (or its modules). For complex simulations based on components, the process of decomposition of the system may often be performed in many ways, depending on factors, such as precision-performance tradeoff. This elementary problem, that every author of simulation always need to face, is a topic of many discussions that can be found in the literature, e.g. in [47, 48, 50], and raised (in general terms) already in [49]. In HEP simulations this decomposition may often be dictated by the laws and phenomenas of physics (such as spin correlations), which may or may not be taken into account in the conceptual (and computer) models, depending on requested (and expected) precision of results. Hence, the choice of the "splitting lines" (thus: definition of a component) may be appropriate or inappropriate, depending on precision and application context.

[6] which is caused by varying requirements dictated by e.g. "irrational" laws of quantum physics, that stand behind new physics models being implemented

The choice of the conceptual model may have dramatic impact on the way in which the computerized simulation model is organised, and the scope of possible (i.e. physically valid) composition contexts. Let me bring the case of TAUOLA here, with its various modes of operation differing by the way in which spin information is treated. In the "generic-purpose", mode, spin correlations are either neglected completely or treated with approximations (in the TAUOLA Universal Interface), yet TAUOLA may be used in a "standard" HEP simulation architecture, with simulation modules being called subsequently (i.e. allowing τ's to be produced in one module, then, to generate their decay, independently, by TAUOLA). When the model with full spin treatment is chosen, however, production and decay processes cannot be separated, and a special organisation for the interface between the "host" generator (such as KKMC [55]) and TAUOLA needs to be arranged.

To illustrate the importance of the precision of the model implemented by the component, let me present another example, of the PHOTOS Monte Carlo, the precision of which, for the general use case, was not guaranteed to exceed "crude" level, suitable for instance for full phase-space detector coverage studies. In the studies presented in [69], PHOTOS is compared with a first-order generator (KLOR) for $K^0_{l3} \to \mu\pi\nu(\gamma)$ decays, and significant discrepancies are concluded. The version of PHOTOS of that time, however, was known to provide only "crude level", hence the comparison was not meaningful. The article [69] motivated us to implement (what turned out to be trivial) correction weight (responsible for interference of emission from the two charges), and this way the new version of PHOTOS, starting from version 2.09, had already higher precision level. This topic is further discussed in section 18.2.4.

In the definition of a component for HEP simulation the mandatory requirement of "known (documented) properties" for a component gains broader sense. Not only the algorithm and the component needs to be documented at the technical level, but also its physical contents and precision need to be assessed. The approximations used in the conceptual model, the range of their validity and limitations need to be specified. These issues lead to the question of the systematic uncertainty of the model, which is a crucial factor for determining the ultimate precision in the data analysis use cases. The approximations used in the model are of two types. In the conceptual model, the approximations such as "soft photon limit", or "leading logarithms" are related to theoretical uncertainties or the way in which analytical calculations are performed. In the computer model, numerical approximations may also be non-negligible, and need to be assessed properly.

The idea of component-based programming is often presented in a simplified picture of a "black-box". Despite being quite effective in commercial use, this naive approach may be dangerous when used with components containing rich physical content. Such component should not be used "blindly", without at least elementary understanding of physical process it models. In particular, when component is composed in a new environment, the validity of approximation always need to be assessed, which usually requires understanding of physics. It should also be known what kind of results to expect from the component.

Ultimately, the tradeoff between performance and error-proofing need to be chosen. From my practise I conclude that components may be used in contexts that their authors would never thought of (e.g. the use of PHOTOS in astrophysics calculations). This implies that either the interfaces or data-input routines need to perform at least basic sanity checks on the parameters and input data, and - if needed - be able to gracefully recover from errors. In the case of PHOTOS, this involves checking the grammar of the HEPEVT event record and kinematics correction routines. I also found it useful to incorporate sanity checks, connected with "emergency STOP" behaviour at various stages of the code of the algorithm. Such "STOP" messages, used e.g. in PHOTOS, are priceless for the authors of the code,

because they prompt the users to report encountered problems, and (also) check and verify that the context of use is proper and the physical approximations are still valid. Certainly, putting too much protections or error-recovery may significantly complicate the code, or cause performance penalty. A proper balance needs to be drawn.

Chapter 5

ATLAS

ATLAS (**A** **T**oroidal **L**arge **A**parat**u**s) is a multi-purpose detector constructed at the LHC accelerator at CERN. In this chapter I will provide the reference information concerning its physics programme, detector, trigger and data acquisition system and computing architecture.

5.1 Challenges in elementary particle physics

5.1.1 Current theoretical picture: Standard Model

The Standard Model (SM) is one of the most succesful and precise theoretical descriptions of Nature, characterising the components of matter and their interaction. Its predictions match with phenomenas at the smallest scales (10^{-18}m) and the highest energies (~1 TeV) observed in High Energy Physics experiments available so far. SM is a quantum field theory describing the interactions of spin 1/2 point-like fermions, whose interactions are mediated by spin 1 gauge bosons. The bosons are a consequence of local gauge invariance applied to fermion fields and are a manifestation of the symmetry group of the theory: SU(3) $\times$ SU(2) $\times$ U(1).

There are 12 fundamental fermions, 6 leptons and 6 quarks, the existence of which was confirmed (directly or indirectly) experimentally. Quarks and leptons form three generations, which differ significantly in their masses. The origin of this structure (and the breaking of flavour symmetry) remain unexplained. Each

Table 5.1: LHC parameters [70] and certain process rates/cross-sections.

integrated 3-year luminosity for $L_{initial} = 2 \times 10^{33}\text{cm}^{-2}\text{s}^{-1}$)	30 fb^{-1}
integrated 1-year luminosity for design $L = 1 \times 10^{34}\,\text{cm}^{-2}\text{s}^{-1}$)	100 fb^{-1}
bunch collision rate	40 MHz
Centre-of-mass energy	14 TeV
inelastic events/bunch crossing (design lumi)	23
total crossection, σ_{tot}	100 mb = 10^{-25} cm^2
inelastic crossection	70 mb
total interaction rate, $\sigma_{tot} \cdot L$	10^9 Hz
rate for SM Higgs, $m_H = 120GeV$ in a rare $H \to \gamma\gamma$ channel	0.001 Hz
inclusive crossection and rate for b-quarks(design lumi)	0.6 mb / 6 MHz
rate for inclusive W production (design lumi)	300 Hz

1 barn=$10^{-28}m^2$, $1\,mb$=$10^{-27}cm^2$

of the three generations of leptons contain one massive particle with electric charge (-1) and one electrically neutral, massless particle. The three charged leptons are: the electron (e), muon (μ) and tau lepton (τ); the three neutral leptons are the neutrinos: ν_e,ν_μ and ν_τ. Similarly for the quarks (each of them being massive), each of the three families contain one quark with electrical charge ($+\frac{2}{3}$) (namely the "up" (u), "charm" (c) and "top" (t) quarks), and one quark with electrical charge ($-\frac{1}{3}$) (namely the "down"(d), "strange" (s) and "bottom" (b) quarks). The quarks are triplets under the SU(3) group and thus additionally carry "colour" charge. The mass eigenstates of quarks differ from their weak interaction eigenstates, which leads to the three generations being mixed. The mixing mechanism is parameterised by the Cabibbo-Kobayaski-Maskawa (CKM)) matrix, the origin of which is not explained by the Standard Model.

The SM provides theoretical description for three out four known interactions: strong, weak and electromagnetic. The SU(3) group describes the strong interactions (quantum chromodynamics, QCD) mediated by eight vector gluons. The massless gluons carry colour charge, hence they are self-interacting. This requires that the QCD coupling, α_s, is small for large momentum transfer (which manifests asymptotic freedom of quarks), but large for small momentum transfers (which leads to the confinement of quarks inside colour-neutral hadrons). Experimental manifestation of the strong interaction is the production of jets of hadrons in high-energy particle collisions, caused by quark-antiquark pairs and gluon production in response to an attempt to "free" a quark.

The description of the electromagnetic and weak interactions were unified in a common theoretical model of electroweak interaction, described by the $\mathrm{SU(2)} \times \mathrm{U(1)}$ symmetry group. The group symmetry is spontaneously broken by the existence of postulated Higgs field with non-zero expectation value. This leads to the emergence of massive vector bosons, the $W^\pm$ and Z^0, which mediate the weak interaction, and the massless photon (γ) for electromagnetism. The manifestation of one degree of freedom remaining in Higgs sector should be the existence of a neutral, scalar boson H^0, which has not been observed so far.

The Standard Model of particles and interactions has 19 parameters: the three coupling constants of the gauge theory $\mathrm{SU(3)} \times \mathrm{SU(2)} \times \mathrm{U(1)}$, three lepton and six quark masses, the mass of the Z boson, which sets the scale of weak interactions, and the four parameters which describe the rotation from the weak to the mass eigenstates of the charge $-\frac{1}{3}$ quarks (CKM matrix). The values of these parameters are known, with varying precision. The remaining two parameters are the CP-violation parameter associated with the strong interaction (the value of which must be very small) and the parameter responsible for the breakdown of electroweak $\mathrm{SU(2)} \times \mathrm{U(1)}$ to $\mathrm{U(1)_{em}}$. This parameter may be expressed by the mass of the (undiscovered) Higgs boson, which couplings may be determined once its mass is known. Despite the SM not providing any indications of the mass scale of the Higgs sector, some constraints may be provided by perturbative calculations and current experimental results. If the Higgs mass in in range 160-170 GeV, the SM is well behaved up to Planck scale of 10^{19} GeV; for other values, new physics must exist below Planck scale. The dynamics of ZZ and WW interactions indicate that the Higgs boson mass must be less that 800 GeV, otherwise the dynamics of these interactions will reveal new structure at energies of 1 TeV[1].

5.1.2 Other theoretical models and extensions

The Standard Model, when confronted with experimental results, seems to present an almost complete and self-confined picture of the elementary particles and inter-

[1] This is the argument that sets the scales for the experiments that would probe the nature of electroweak symmetry breaking, such as ATLAS.

actions. The only missing element predicted in the SM, yet not observed yet is a single, neutral scalar Higgs boson, H^0, with mass in range 160-170 GeV. The existence of a single boson is, however, unsatisfactory for many theorists. It is believed that the SM is only a part of a more fundamental theory, with a larger energy scale. For instance, cancellation mechanisms postulated to resolve some calculation of radiative corrections of the Higgs boson mass suggest that new physics should appear on the scale of 1 TeV: new particles or new strong dynamics may appear. Let us shortly present a few of them, which are relevant to the energy scale probed by the LHC.

One of the most elegant and appealing models, with no experimental evidence so far, is Supersymmetry. Not only it provides the way to integrate gravitational interactions into quantum theory of particle interactions, but may also (in some models) allow for unification of coupling constants at some high energy scale, and hence reduce the number of parameters. Supersymmetry retains the whole success of the Standard Model (including the unbelievable precision for electroweak predictions), while providing mechanisms for cancellation of divergences and tunings of the Higgs mass.

In supersymmetric models, the existence of a new family of (currently not observed) particles, in postulated. Each of particles would have a supersymmetric partner: fermions would have bosonic superpartners (squarks and sleptons) and bosons would have fermionic superpartners (gluinos and gauginos); multiple Higgs bosons would also be present: h, H, A, $H^{\pm}$. The masses of these unobserved particles are calculable provided the values for model parameters (not available so far) are given. It is believed that if supersymmetry is responsible for electroweak symmetry breaking, it will demonstrate itself in the region of energies of order of 1 TeV, or smaller.

An example of a model based on dynamical symmetry breaking is "technicolour" for strong coupling. The model predicts a large spectrum of states and a structure in the WW scattering amplitude, in the region of 1 TeV, if the dynamics have anything to do with electroweak symmetry breaking.

Other possibilities for new, "exotic" physics, not related to the scale of electroweak symmetry breaking are new, heavy gauge bosons, quarks, charged leptons, and massive neutrinos, a structure in quarks and leptons, or monopoles.

5.2 ATLAS Physics programme

The ATLAS physics programme is covered in detail in [45]. Here let us shortly present the most important tasks and focus points of this programme.

The main motivation for the ATLAS experiment experiment, which guided the detector design and optimisation, was to perform the measurements leading to an understanding of the mechanism of electroweak symmetry breaking, with a major focus on the Higgs boson, but also other related phenomena such as particles predicted by supersymmetry or technicolour models, new gauge bosons and evidence for composite nature of quarks and leptons. The investigation of CP-violation in decays of B hadrons (so called "B-physics") and the precision measurements of W and t-quark masses and triple gauge boson couplings will also be important components of the ATLAS physics programme.

At the initial phase of LHC running with low luminosity ATLAS will work as a "factory" for QCD and heavy flavour processes and gauge boson production, to allow various precision measurements be performed already in the early stages of the experiment.

The QCD related studies will profit from the availability of the high-statistics data for the new energetic regime, never probed before. Jet and photon physics,

charm, beauty and gauge boson production, as well as diffractive processes will be explored. Precise constraints for parton distribution functions (PDFs) will also be derived; deviation from the QCD theoretical predictions might already indicate the onset of new physics, such as compositeness, at that stage. Measurements and understanding of QCD processes will also be essential as they form the dominant background in searches for new phenomena.

Despite the B-physics is not the main field of study for ATLAS (another LHC experiment, LHCb, is dedicated to such studies), it will profit from the LHC acting as "b-factory", with 10^{12} $b\bar{b}$ events expected per year at low luminosity. This high statistics will only be limited by the maximum data recording rate, therefore a variety of specific measurements related to CP-violation, B_s^0-mixing, rare mesons decays and B-hadrons spectroscopy will be performed. ATLAS can perform competitive high-precision measurements of these processes. ATLAS will also take advantage of high LHC luminosity to perform t-quark precision measurements and studies (8 million of $t\bar{t}$-pairs expected to be produced in one year of low-luminosity run (10 fb^{-1})). The mass of the W boson is going to be measured with precision better than 20 MeV (expected statistical uncertainty for 300 million single W events expected in one year data taking will be about 2 MeV!), to ensure that the mass of W is not the dominant source of errors in predictions for the SM Higgs mass. The available large rate of gauge boson pair production will allow to provide critical tests of triple gauge-boson couplings, probing the underlying non-standard physics.

The search for the Higgs boson and its supersymmetric extensions in the Minimal Supersymmetric Standard Model (MSSM) will be the main focus of activity. ATLAS is sensitive to a variety of possible signatures, being accessible at the low or at the design luminosity. The Higgs boson (or multiple Higgs bosons) is expected to be detected in one of the $h \to \gamma\gamma$, $h \to b\bar{b}$, $H \to ZZ \to 4l$, $H/A \to \tau\tau$, $H/A \to \mu\mu$. The clearest signatures leading to recontruction of narrow mass peaks are in the photonic or leptonic decay channels, but other multi-jet and multi-τ channels are also promising. The expected signal-to-noise ratio for many cases are much smaller than one, hence the detection of the Higgs boson will be an experimental challenge. ATLAS will cover the full Higgs mass range up to 1 TeV for the SM Higgs and the full parameter space for the MSSM Higgs; it has also a large potential for searches in alternative scenarios.

The detection of Supersymmetric particles at ATLAS should be straightforward if it exists at the electroweak scale. The expected cross-sections for the production of heavy (1 TeV) squarks and gluinos should be as large as a few pico-barns. Their cascade decays should lead to signatures allowing for precision measurement of the masses of supersymmetric particles and determination of model parameters. The strategies for searching other "exotic" physics phenomena, such as technicolour, excited quarks, leptoquarks, new gauge bosons, right-handed neutrinos and monopoles were also prepared.

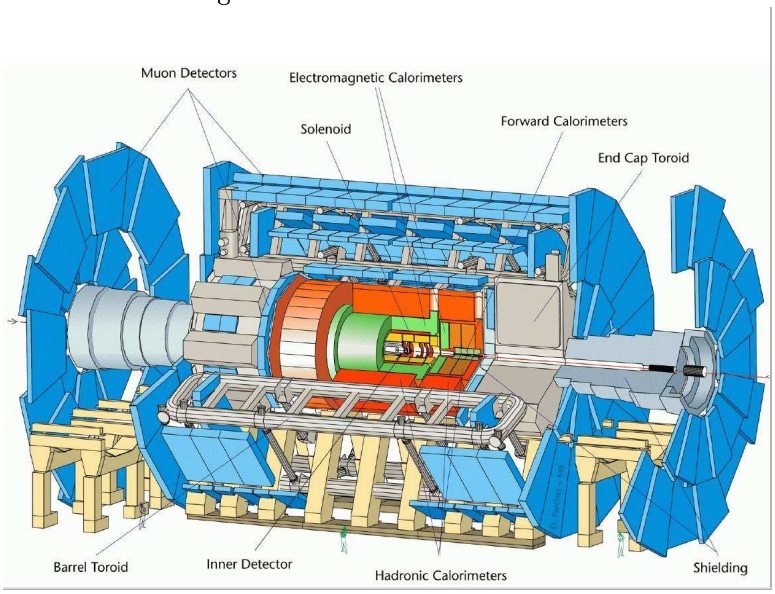
5.3 ATLAS Detector

5.3.1 Physics requirements

The design and optimisation of the ATLAS detector were performed according to the main points of the physics programme, sketched in Subsection 5.2, namely the exploration of the electroweak symmetry breaking mechanism. Let us now summarise the requirements for the performance of the detector that would assure the achievement of the ATLAS physics goals.

Higgs boson searches in the full range of allowed mass requires sensibility to most challenging signatures, therefore these requirements were used as a bench-

Figure 5.1: The ATLAS detector.



mark for setting the parameters of the detector performance. The most important are: high resolution of measurements of electrons, photons and muons, excellent secondary vertex detection for τ-leptons and b-quarks, and high-resolution calorimetry for jets and missing transverse energy (E_T^{miss}).

Supersymmetry searches set the requirement on the hermeticity and E_T^{miss} capability of the detector, and on b-tagging at high luminosity.

New heavy gauge bosons searches establish benchmark requirements for high-resolution lepton measurements and charge identification in the p_t as large as a few TeV.

Quark compositeness signatures set the requirements for the measurement of very high-p_t jets.

B-physics programme and precision measurements: the determination of precise values for the masses of W and t-quark, studies of gauge boson couplings, CP-violation and CKM unitary triangle establish the requirement for precise control of the energy scale for jets and lepton, precise determination of secondary vertexes, full reconstruction of final states with relatively low-p_t particles and triggering on low-p_t leptons. Physical requirements of B-physics (namely the determination of secondary vertexes) establish technical benchmark for the High Level Trigger and Data Acquisition (the full scan of the Inner Detector may be required at LVL2 trigger).

5.3.2 Layout of the ATLAS Detector

The rich physical program and physics requirements of the ATLAS experiment require the construction of the state-of-the-art, complex particle detector for "general-purpose" use. The scheme of the ATLAS detector is presented in figure 5.1.

The overall concept of the ATLAS detector is a typical one, with high-resolution multi-layer particle tracking system near to the interaction point (Inner Detector, ID), followed by subsequent layers of calorimeters, and a magnet system bending the tracks of charged particles. The characteristic element of ATLAS is its impressive

muon spectrometer, which involves the use of toroidal magnetic fields and huge detectors that forms the outer-most shell of the detector; ATLAS is the largest detector built for the LHC. Let us now briefly present all the elements (subdetector) of the ATLAS detector.

The Inner Detector is responsible for providing high-resolution information about the tracks of the particles, in the most important region next to the interaction point, where vertex resolution is essential. It is built of three subsystems: two semiconductor detectors: pixel and silicon microstrip (SCT), and the straw tube tracker (transition radiation tracker, TRT). The highest granularity is achieved around the vertex region where pixel detectors (with $R\phi$ resolution of 12 μm) are located; 3 layers (including one removable, replacable "B layer", closest to the interaction point, that is mostly endangered by defects due to high radiation doses) in the barrel section and 5 end-cap disks on each side provide 140 milion readout channels covering the region of η up to ± 2.5 (2.3 m^2 of area). Very high resolution provided by the pixel dubdetector is vital for resolution of impact parameter and identifying short-lived particles such as B-mesons or τ leptons.

The pixel layers are followed by SCT layers: 4 in the barrel region and 9 end caps on each side; they provide $R\phi$ resolution of 16 μm and 6.4 milion of readout channels and cover the region of η up to ± 2.5 (over 61 m^2 of area). Typically, three layers of pixels and eight strip layers (four space points) are crossed by each track.

The outermost region of the Inner Detector is composed of TRT detectors, arranged axially in the barrel region and radially in the end cap disks. Despite the resolution of a single TRT straw is much worse than that of silicon detectors, 170 μm, it provides a large number of tracking points - typically 36 points per track. Strong advantages of the TRT detectors over semiconductor detectors are significantly smaller cost and much less material introduced. Additionally, TRT detector is capable of providing electron identification by the detection of transition radiation photons produced in the radiator material between the straws. The combination of straw-based techniques at the outer radius of ID, with semiconductor high-resolution technologies at internal makes a very robust tracking device. The high number of hits registered by TRT compensates for its smaller precision (with respect to semiconductor tracker)at the outer radius and provides the way for accurate momentum measurement.

To resolve the momenta of charged particles the Inner Detector is put in the 2 T magnetic field generated by superconducting Central Solenoid (CS) magnet. Because of the CS position "in front of" electromagnetic calorimeter, and the need to minimise the material introduced by CS in order to achieve desired calorimeter performance, the CS and the LAr calorimeter share the same vacuum vessel.

The ATLAS calorimetric system is composed of electromagnetic calorimeter (EM), and a set of hadronic calorimeters that employ differing detection methods in the barrel and endcap regions. The EM calorimeter is a lead-liquid argon (LAr) detector, with unique, "accordion" geometry which provides complete ϕ symmetry, and optimisation of the width of absorber as a function of pseudorapidity enhancing energy resolution. A thin "presampler" layer is used in the regions where the amount of up-stream material is significant (exceeds $2X_0$) to correct for the energy loss of photons and electrons, and also enhancing particle identification ($g/\pi^0, e/\pi$). The barrel part of the EM calorimeter is built of two identical sub-barrels and covers the region of $|\eta| < 1.475$. Each of endcap calorimeters is also divided into two coaxial wheels: outer covering the region $1.375 < |\eta| < 2.5$ and inner wheel covering the region $2.5 < |\eta| < 3.2$.

ATLAS hadronic calorimeter uses two different techniques: iron scintillating-tile sampling tile calorimeter for the barrel at $|\eta| < 1.7$, and LAr calorimeters for the range of $1.5 < |\eta| < 4.9$, in the endcap calorimeter (HEC) and high-density forward calorimeter (FCAL). An important parameter of the ATLAS hadronic calorimeter

is good containment for hadronic showers, so as to reduce "punch-through" into the muon subsystem downstream. The total thickness is 11λ (interaction lengths), which effectively reduce the impact of this effect. A good resolution for high-energy jets and large pseudorapidity coverage assures good E_T^{miss} measurement as well. Each HEC is built of two wheels, which in turn contain 32 modules and weights over 150 tons.

The outer part of the ATLAS detector is the Muon Spectrometer. It is based on the magnetic deflection of muon tracks in the large, superconducting air-core toroid magnet, instrumented with separate trigger and high-precision tracking chamber. The toroid magnet system is composed of a barrel toroid (BT) and two endcap toroids (ECTs); each of these three toroids consists of eight coils, assembled radially and symmetrically around the beam axis. The total weight of the toroid magnet system exceeds 1200 tons; the current flowing through the 24 coils is 21 kA. The configuration of the toroidal magnets provides the field that is mostly orthogonal to muon trajectories, while minimising the degradation of resolution due to multiple scattering. The detectors used in the Muon Spectrometer use four different chamber technologies, arranged in layers (called stations): Monitored Drift Tubes (MDTs), Cathode Strip Chambers (CSCs), Resistive Plate Chambers (RPCs) and Thin Gas Chambers (TGCs). Large size of the system (and its elements) makes it impossible to assure mechanical rigidity and precision on the level of $30\,\mu m$ (typical resolution of chambers), therefore optical alignment measurement facilities need to be integrated, to introduce the corrections (due to mechanical displacements up to ~1 cm) to offline physics analyses.

5.4 ATLAS Trigger and Data Acquisition System

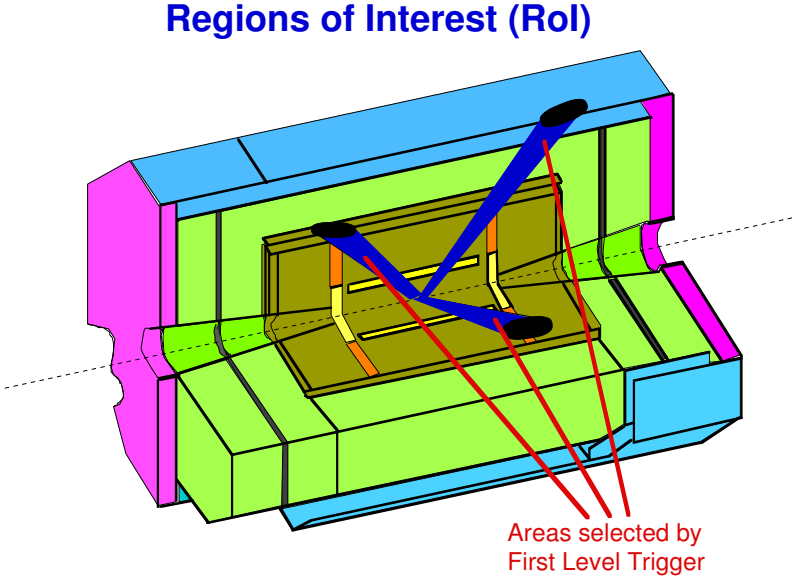
At the design luminosity of $10^{34}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$, and bunch-crossing rate of 40 MHz, the rate of interactions occurring in the LHC experiments will be of order of 10^9 Hz. Due to complexity of the detector, and the size of data (for raw data representing readout of complete detector it is around 2 megabytes), and technical limitations in the throughput of data storage facilities, the rate of event saved in permanent storage needs to be reduced to ~100 Hz[2]. The event filtering (trigger) and data acquisition system of excellent efficiency is a key subsystem for each of the LHC experiments: while the 10^7 rejection factor against minimum-bias event need to be achieved, the signature of rare physics processes (e.g. the decays of Higgs boson) must not be lost.

The ATLAS Trigger and Data Acquisition system, described in length in [71, 70], is going to achieve its goals by means of three stages of event rate reduction facilities: the LVL1 trigger, the aim of which is to reduce the initial rate of events to 75-100 kHz, working with time budget of $2.5\mu s$ (defined by bunch crossing rate) , LVL2 trigger reducing the rate to ~1kHz, with decision latency of order of single milliseconds, and the Event Filter (EF), which will reduce the event rate to ~100 Hz within single seconds. Due to extreme requirements for the latency, the LVL1 Trigger is constructed using custom, programmable, high-speed electronics, closely integrated with detector's front-end electronics. The LVL2 and EF, refereed to collectively as High Level Trigger (HLT), are in turn closely related to the data collection and full event building (Data Flow System) and build using commodity PC/Linux based computers, executing highly optimised data analysis algorithms, and large, switched Ethernet networks.

The LVL1 trigger performs an initial selection of events based on reduced-granularity information, from a subset of detectors. The decision is based on quite

[2]Even at that rate, each of the LHC experiments will require petabytes of storage space per year, only to store unprocessed data.

Figure 5.2: Regions of Interests (ROI) guide LVL2 trigger algorithms



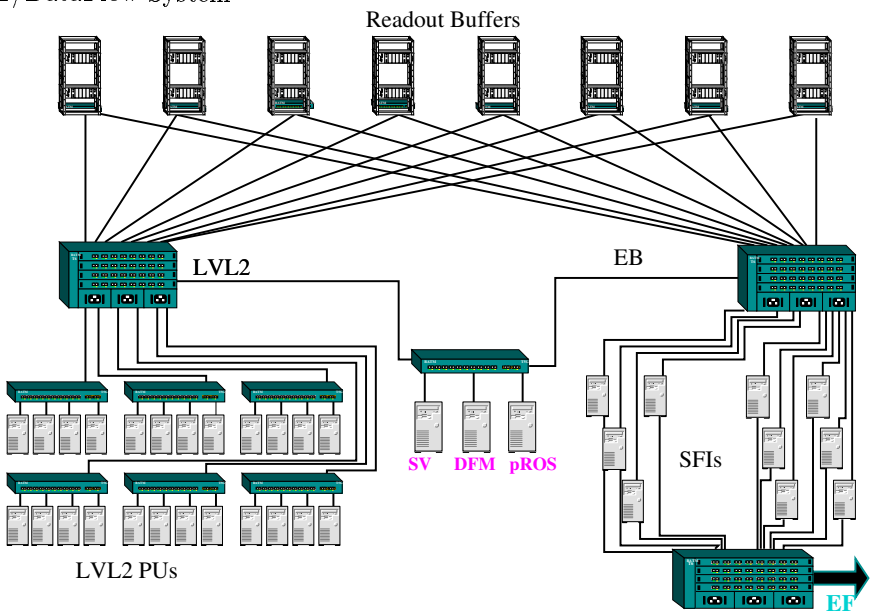
simple, inclusive selection criteria of combination of objects required in coincidence. During the time the LVL1 decision is being taken the complete data of detector read-out systems (for over 10^7 channels) need to be conserved in buffering pipeline memories; once the acceptance decision is taken, the data is read-out from frontend electronics and stored in RODs (Read Out Drivers), then copied to around 1600 ROBs (Read Out Buffers), from which it is available for the High Level Trigger. Together with the decision of the event being accepted, the LVL1 trigger works out the information about RoI's, Regions of Interest (see Figure 5.2), which is then used as initial guidance for the LVL2 Trigger algorithms.

The operations that need to be performed by High Level Trigger and Data Flow systems of ATLAS are closely related, and depend on each other. The responsibility of the Data Flow system is to provide the LVL2 and EF trigger algorithms with completed information concerning an event: in the case of LVL2, partial readout information is requested for specified geographical regions of the detector, whereas for the EF the complete information of a single event needs to be gathered and assembled in the process of event building. Event building is performed, however, only for the events that were accepted by the LVL2 trigger.

The architecture of the ATLAS High Level Trigger and Data Flow systems were a subject of my scientific activity as CERN Doctoral Student in 2001-2003 (which is described in part *II* of this thesis); the information contained in this section is supposed to serve as a source of reference documentation.

The ATLAS HLT and Data Flow system has a form of cluster-like farm of PC computers, connected with each other and with around 1600 Read Out Buffers by a large Local Area Network (LAN). The data from the ROBs (or clusters of them, called ROS's) is transmitted via the network to hundreds of LVL2 Trigger Processing Units (L2PU), and around 100 SubFarm Interfaces (SFI), which build complete events, and pass them to the Event Filter processing farms. The flow of data and load balancing is controlled by a few LVL2 Supervisors (L2SV), and one Data Flow Manager (DFM); the decision worked out by the LVL2 is stored in dedicated Read Out Buffer called pseudo-ROS (pROS), which is read out during event building. In Figure 5.3 a scheme of a possible implementation of the system, with all the components, is presented.

Figure 5.3: A possible implementation of the network architecture for the ATLAS HLT/DataFlow System



The system works in the following way: after an acceptance decision of the LVL1 Trigger (rate 75-100 kHz), the data read out from the detector are copied to the Read Out System (ROS), which is built of more than 1600 Read Out Buffers (ROS); the results of LVL1 processing are also transfered to the LVL2 Supervisor. The Supervisor selects a LVL2 Processing Unit using a load-balancing approach, and assigns the processing of the event to it - the information of the Regions of Interests is transferred as well. The LVL2 PU, which may process multiple events in parallel, starts the processing of a new event as soon as it has resources to do so. The LVL2 Feature Extraction algorithms are executed sequentially, and usually require that data is read out from one or many ROBs in every step; LVL2 firstly verifies the information passed by the LVL1 trigger in the initial set of RoIs, then localises additional region of interests. The number of processing steps and data buffers that need to be requested cannot be therefore known a priori. Once the LVL2 decision is taken (accepting or rejecting an event), it is transferred to the Data Flow Manager; in the case of accepting decision, the details of LVL2 processing are stored in the pROS. Upon reception of a reject decision, the Data Flow Manager (DFM) sends the message to the ROBs to remove the rejected event from the buffers. In the case of the "accept" message, the DFM selects an idle SubFarm Input node (SFI) and orders it to build selected event; if no idle SFI exists, the event is queued in the DFM (may also be queued in the SFI). After receiving event assign message, SFI starts to query the data from all the ROBs, and assemble the complete event data; once ready, it sends the confirmation to the DFM, and the complete event to the SubFarm Output (SFO), which in turn assign it to Event Filter processing farm. When DFM receives confirmation from the SFI it may again send (grouped) clear message to all ROBs. The expected total rate of data flowing from the ROBs to the L2PUs and SFIs is about 5 GB/s. The rate of event build does not exceed a few kHz. The Event Filter nodes firstly confirm the LVL2 decision, then execute refined selection and event analysis algorithms on the full event data (without callibration though); certain time-consuming algorithms (such as vertex and track fitting using bremsstrahlung recovery for electrons) are possible to execute only at that level.

Full events accepted by the event filter are recorded at the rate of ~100 MB/s.

5.5 ATLAS Off-line software: ATHENA

Computing infrastructure, the key component in the data reconstruction and analysis for ATLAS is described in detail in [42], hence it will only be briefly presented here. Enormous amounts of experimental data that needs to be processed and made accessible for physicists performing their analyses around the world motivated tier-based solution and distribution of the data between computing centres; GRID technology will be sported to provide the middleware infrastructure related to data storage and replication, and the management of computational resources and computational tasks.

ATLAS software for off-line analysis is written in the component-based C++ framework called `ATHENA` [57, 62, 61], which is a concrete implementation of the `GAUDI` component architecture [59, 51] designed for use in physics data-processing applications. The `GAUDI` architecture, initially developed in LHCb, is now a common effort of ATLAS and LHCb, used also by other experiments, such as HARP, or GLAST. The component approach of `ATHENA` is based on a "blackboard model"[3] for transient objects (which may also be made persistent), and allows to reduce the number of the interfaces that need to be implemented by components. Following the concepts of GAUDI, ATHENA introduces a clear separation of data and algorithms (producers of the data). `ATHENA` uses `HepMC` to store the data produced by event generators; the model of the this event-record data storage does not seem to be compatible with the transient data store ("blackboard model") of `GAUDI`, causing performance problems.

`ATHENA` (and `GAUDI`) seems to be a good example of the use of component-based approach to software, even though its power is limited in certain areas, such as event generation, where functional, reliable C++ event generators does not exist, and the interfaces to FORTRAN codes (such as [72]) need to be provided. The management of components responsible for multi-step process of data reconstruction and analysis is certainly a success of `ATHENA` in the domain related to the experiment. Unfortunately, the boundary with theoretical physics which is nowadays often expressed by means of Monte Carlo event generators does not seem to be approached at all: an "ad hoc" solution based on the `HepMC` data structure shows its limitations, and stresses even more the need for a reliable, commonly agreed (amongst both: experimental and theoretical physicists) standard of the event record in C++. Part *III* of the thesis is dedicated to the discussion of this topic.

[3] StoreGate being the ATLAS implementation of the Transient Data Store

Part II

at2sim: ATLAS Data Flow System simulation

In memory of
dr Robert W. Dobinson
(1943-2004)

Chapter 6

at2sim: Introduction

Overview In this chapter I will firstly discuss the motivation and the early implementation of the `at2sim` model; then, in section7, I will present the results of the study of the Linux networking routines, stressing their influence on the design of the model. In section 9.3 I will present the architecture model of a generic end-node, and shortly present the models of the Data Flow applications. In section 9.4 I will report on callibration of the models performed on small setups. In section 9.5 I will present the results of modeling efforts.

Motivation for computer modeling

The ATLAS Dataflow System will be built of a few thousands components, interconnected by a large Local Ethernet Network. Because of the complexity of the system and its costs it is not possible to construct a full-scale prototype. The scaled-down prototypes, up to the scale of 10 %, were built and tested, mainly to demonstrate the functionality of the components and the possibility of construction of the final system out of them. These prototypes, however, cannot give the definite answer to the questions of the scalability to the full size, and the performance of the full-scale system. The answer based on simple approximations certainly does not give enough confidence.

In the first approach, a system might be designed upon the knowledge of requirements for the data throughputs and rates, coming from physical models of interactions, estimated process rates and the design of the detector. The so called "paper model" for the ATLAS Dataflow System realises this approach, and gives the estimates for average rate and processing requirements. The results given by the paper model for various running condictions of the LHC are available in the Appendix A of the [70].

The sole availability of sufficient amount of network bandwidth and computing resources cannot, however, guarantee that the required performance requirements are met. The dynamic nature of the data-taking process, and the constraints imposed by real-time regime of this operation requires that the following issues are also assured:

- the computing load should be evenly distributed to the available computing resources
- the data buffers of all communicating elements should be large enough, congestion should be minimal
- the system should be designed to gracefully cope with realistic fluctuations in rates and throughputs, by providing sufficient spare processor and network resources.

To answer the question of the system scalability, taking into account the aforementioned issues, the dynamic behaviour of the system needs to be studied. Again, due to the complexity of the system, the only viable method is computer simulation using a "computer model" of the full-scale system. Two dynamic computer models were constructed for the ATLAS Dataflow System: `at2sim` and `Simdaq`.

The model of the dataflow system implemented in both tools is an object oriented model, in which most objects represent hardware (e.g. switches, computer links, processing nodes), software (e.g. low level network communications in Linux OS, Data Collection applications) or data items (e.g. Ethernet packets). The focus of the `at2sim` program was initially a proper simulation of the event building, while in `Simdaq` details more relevant to the LVL2 trigger are taken into account (in particular proper handling of the step-wise execution of the trigger algorithms and associated requesting of event data from the ROBs).

The type of simulation used for the computer models in known as "discrete event simulation". Basically, the simulation program maintains a time-ordered list of "events" i.e. points in time at which the simulated system changes state in a way implied by the type of "event" occurring. Only at the time of occurrence of an event is the modeled system allowed to change its state; in most cases only a small part of the state of the simulated system needs to be updated. The state change can result in the generation of new events for a later time; these events are then entered at the correct position in the event list. The simulation program executes a loop in which the earliest event is fetched from the event list and then handled.

The development of the at2sim model was one of the main areas of scientific activity during my CERN Doctoral Student stipend, in years 2001-2003. In this chapter I shall describe the architecture of this model, and the process of its parameterisation. I will also partially present the results of the model for the final system. Even though I quit the work on the `at2sim` project in Spring 2003, th model, is still being used to build up the confidence in the design of the ATLAS Dataflow System and provide new estimates .

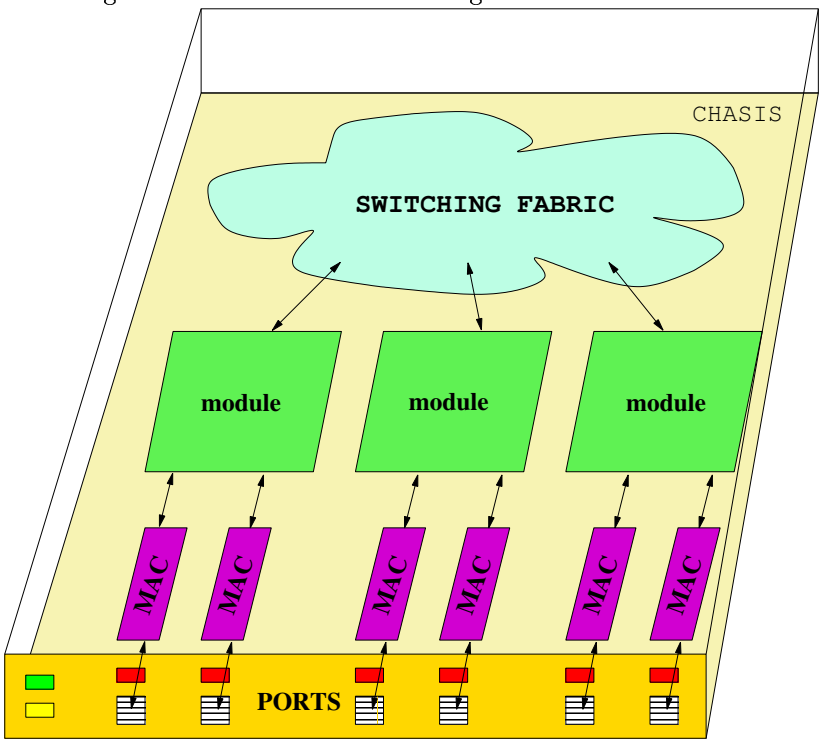
First implementation and the motivation for the "new" at2sim

In February 1998, the "LVL2 Pilot Project" [73] was started, to prepare the material for the "Technical Proposal" document [71]. The goals of the project were: to demonstrate the full LVL2 trigger functionality, with rejection factor of at least 100, to contribute to the complete High Level Trigger architecture and prepare the integration with the Event Filter, and to select the technology candidates. From the practical point of view, that led to the preparation of functional components, software testbeds and demonstrating the system integration. One of the important activities defined in the "System Design" category was the modelling, using the "paper" and the computer models. The `at2sim` (**AT**las lvl**2** trigger **SIM**ulation) model, based on the `Ptolemy` modeling framework [74], was started to be built in parallel with the other deliverables of the project: the "Reference Software" [75] and various technology evaluation tests.

The first version of the `at2sim` model, documented in detail in [76], was the fruit of the effort of a ATLAS T/DAQ modelling group. The effort was put in the simulation of the Pilot Project Ethernet Testbed program, with some efforts to produce a full system model. The design intention was to keep the model as simple as possible, by use of component approach: each node, representing a functional element of the whole system, was seen as a "black-box", which receives simple messages, and, as a result, send out other simple messages at later times, according to transfer functions. The details of the transfer functions, such as delays or message queues were hidden in the details of implementation.

The most important deliverable of this stage of the project was the parameterised

Figure 6.1: The architecture of a generic Ethernet switch



model of a generic Ethernet network switch, presented in [77, 78]. The architecture of the model is presented in Figure 6.1, and the parameterisation in Figure 6.2.

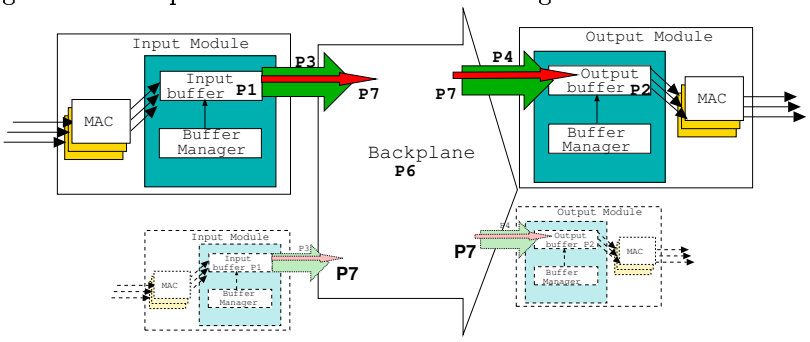
On the basis of this generic model, the ad-hoc models of real switches could be built, by simple object-oriented inheritance technique, accompanied by the appropriate parameterisation. The procedure for determining the values of the parameterisation was also established, which led to the creation of the whole family of parameterised models of real Ethernet switches in Fast- and Gigabit-Ethernet technologies.

Apart from the models of the switches, the complete `at2sim` model contained also the model of message-passing and the models of end-nodes: the supervisor, the emulators of processors and the emulators of ROBs (read-out buffers). It also contained an "event generator", which generated random LVL1 trigger decisions, according to the rates defined in user-defined trigger menus, and the mapping of the ROBs to the (η, ϕ) regions of the detector.

The models of the end-nodes were created to mimic the behaviour of the corresponding components of the Reference Software. They were parameterised using simple, linear response function: upon the reception of a message, a constant (defined by a parameter value) delay was introduced, then an associated reply message was sent. Due to the simplicity of the approach it was very straightforward to measure the values of the parameters for the models, using time-stamping of the code executed on testbed setup. The execution of the models of the nodes was therefore based on a system of queues and delays, and no effort was put to implement the effects of possible parallel execution of threads, etc. The aim of the model at that time was to demonstrate the functionality of the architecture and the Ethernet-based message-passing, rather than to perform precise simulation of timing behaviour of the whole system.

In parallel to the modeling effort, the study of network communication protocols

Figure 6.2: The parameters of the model of the generic Ethernet switch.



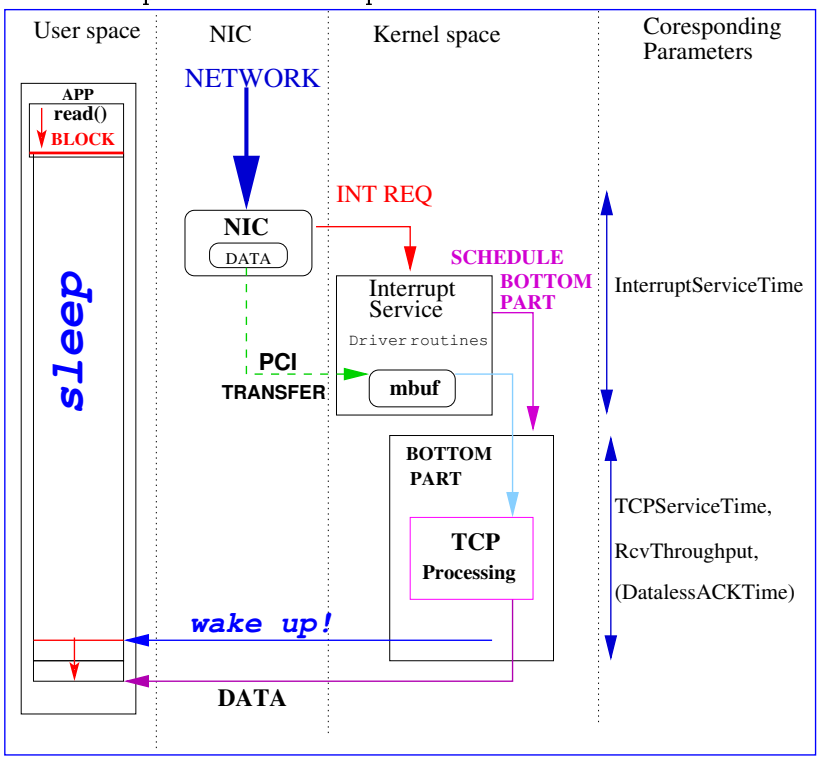

```
P1 = Input buffer length (#frames)
P2 = Output buffer length (#frames)
P3 = Max to backplane throughput (MBytes/s)
P4 = Max from backplane throughput (MBytes/s)
P5 = Max throughput for intra-module(MBytes/s (not shown)
P6 = Max backplane throughput (MBytes/s)
P7 = Inter-module transfer bandwidth (MBytes/s)
P8 = Intra-module transfer bandwidth (MBytes/s)(not shown)
P9 = Inter-module fixed overhead (not shown)
P10= Intra-module fixed overhead (not shown)
```

have been started in the CERN EP/ATR group. At this point I joint the group, as CERN Doctoral Student, and started to take part in the measurements and the modeling of the end-nodes, with the aim to improve the precision of their parameterised models. The results of these early studies, using the Fast Ethernet technology, were described in [79]. Initially, a lot of effort has been put in the study of the TCP/IP protocol, which is the most widespread network communication protocol. It is designed in such way, that it ensures the delivery of the data to the other communication endpoint, detects and re-transmit lost packets, etc, by means of complicated handshake and timer algorithms. We tried to develop a model of a generic computing node, which communicates with another node using the TCP/IP protocol. As a result, we developed a prototype "TCP Broker" model, the diagram of which is presented in Figure 6.3. The model TCPBroker and the enhanced model of the switch were presented at the Real Time 2001 conference [80]. The examples of comparisons of measurement and modeling results for the TCP Broker are presented in Figure 6.4.

The success of the functional model of the Pilot Project gave us the confidence in our approach to the modelling and to Ptolemy-based system and convinced us to discontinue the commercial `OPNET` modelling tool, due to high costs of licenses and its problem with scalability . However, the undebatable success of the model of the Ethernet switch in the `at2sim` was not followed by the models of the end nodes, the functionality and the parameterisation of which were very simplistic. We realised that in order to bring the level of precision of the models of nodes to the level of precision of the switch, we need a major study, design and implementation effort, that would need to completely new implementation of the models of the end nodes. The following arguments were supporting this decision.

- The "Reference Software" was a project aimed at demonstration of the viability of the proposed architecture. At that time, the Technical Proposal [71] document was accepted and the implementation of the real Data Collection Software has started. All the applications running on the end-nodes were being implemented from scratch, based on a common infrastructure for message-passing, control and configuration, following the experience gained

Figure 6.3: The block diagram presenting the model of the operating system activity related to the reception of a network packet.



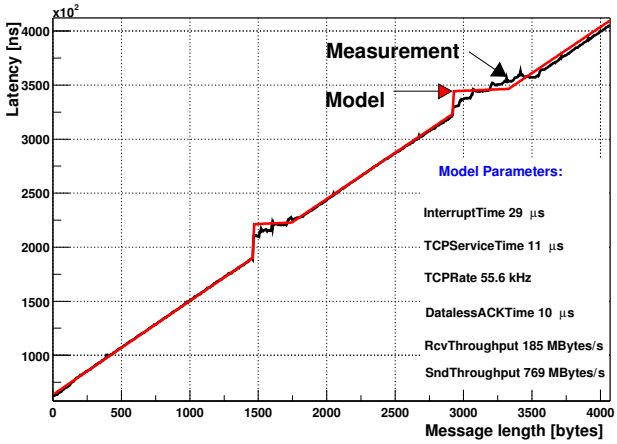
during the Pilot Project.

- The new Data Collection (or Data Flow) software used common message-passing libraries, which enabled the communication over UDP/IP and "Raw Ethernet" protocols. Unlike the TCP/IP protocol, the UDP/IP and Raw Ethernet support multicasting - sending the same message to multiple recipients. The multicast feature was required e.g. for sending the "ROB Clear" messages. In the case of the TCP/IP, which was also available, the multicast functionality had to be emulated using unicast data transfers directed individually to each recipient (TCP/IP is a point-to-point protocol). Because of that, and the additional, un-controllable, load of acknowledgement packets, the TCP/IP was not favoured as the protocol used in the final system.

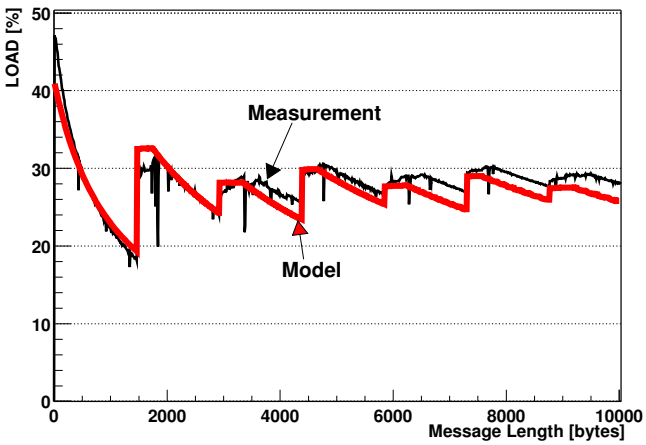
- Unlike the TCP/IP, the UDP/IP and Raw Socket protocols does not include flow-control, packet loss and retransmission mechanisms. The study had to be deployed to ensure that the system run in the regime in which the data is not lost. The typical reason for a data loss is the overflow of some buffer, due to congestion. The study for the full system could only be conducted using the computer model: the possible congestion points in the system had to be identified and eliminated wherever it's possible. To be able to reliably modell the congestion, the simplified model of the queues and resources of the Linux operating system had to be studied and developed. The Ethernet Flow Control mechanism also had to be studied, and the decision on its use determined using the modelling, hence the model of the Ethernet Flow Control had to be included in the model of the switch and the models of the end-nodes.

The implementation of the new models of the end-nodes for `at2sim` was the main

Figure 6.4: The results of early at2sim model of the protocol stack.



(a) Parameterization of the model and the comparison of measured and modelled packet latency.



(b) The CPU load induced by receiving sustained stream of packet: comparison of model and measurements

deliverable of my CERN Doctoral Student stipend in the CERN EP/ATR group.

In the design of the "*new* `at2sim`", I also considered the following arguments:

- The performance of many nodes is determined by the overhead for communication. It would be worthwile to explore the network communication mechanisms of the Linux kernel and understand where the time and resources are spent
- It is the availability of the CPU resources that determines the timing of the events in real systems. In the system with multiple (real or virtual) processors, the system performance does not scale linearly: certain operations (in particular: certain part of network communication routines executed in the Linux kernel) require the use of locking mechanisms, and prevent parallel execution. It is also desirable to have an estimate of CPU utilisation on all nodes in the system, to be able to identify lacking resources and experiment with variations of the architecture.
- The Data Collection applications are multi-threaded, to take advantage of multi-processor systems. This multi-threading behaviour and the sharing of computing resources should be modelled appropriately, if one wants to obtain realistic event sequences, traffic patterns and timing.
- All the end-nodes use common message-passing layer of the Data Collection software. It would be worthwile to follow this approach and separate the parameterisation of the network communication from the parameterisation of the applications' activities.
- The TCP/IP protocol is too heavy to model in detail; we should rather concentrate on the UDP/IP and Raw Socket protocols, favoured by the new Data Flow System.
- The Gigabit Ethernet technology became an "off the shelf" component. The new system should model the hardware reliably, in particular, the CPU-offloading techniques like the interrupt coalescence or NAPI should be studied and modelled appropriately.

Chapter 7

Linux message-passing study and measurements

7.1 Linux kernel mechanisms

The majority of the elements of the ATLAS Data Flow system will be built using commodity "off-the-shelf" computers running the Linux operating system. The TCP/IP protocol stack was an integral part of the Linux kernel since its very beginning. To understand the mechanisms and actual data flow for network communication we needed to understand the core mechanisms implemented in the kernel.

Linux is a multi-threaded multi-processor operating system kernel, based on POSIX standard specifications. It takes care of the tasks like: memory management, sharing and scheduling resources between concurrently running processes, communication with hardware, file-system and network-socket operations, etc. We are not going to give the complete overview of the Linux kernel here, we will however concentrate on the mechanisms related to the network communication.

Receive process

From the application programmer point of view, all the network operations are performed by means of an abstract construct called a *socket*, which conceptually resembles a file descriptor. Once a socket is *open* (i.e. the `open()` operation on the socket was successfully), it is possible to perform *read* and *write* operations. We will start with, the more complex, process of receiving data from the network. The application programmer uses the `recv()` function to receive the data from the open socket. Usually, no data is available yet on the socket, hence the kernel scheduler puts the process that called `recv()` asleep, until the data is received; alternatively, in so called "non-blocking" mode of operation, the `recv()` function returns immediately with the indication of data being not available yet. From now on, we assume that the process was put asleep and waits to be woken up by the kernel (i.e. scheduled for execution by the system scheduler) when the data becomes available.

Whenever a packet is received by the network interface (the NIC), it is put in a local memory of the card. It is however not yet available for the system kernel and the CPU - the data needs to be transferred over the system bus (i.e. PCI) to the main memory of the system. At this stage, the modern cards usually initiate a DMA operation (data transfer without the assistance of the main CPU) to a well define region of the system memory, then it signals the processor, using hardware interrupt, that a new data is available. Upon the reception of the interrupt signal

(unless the interrupts are disabled, e.g. because other interrupt with higher priority is being handled), the processor abandons its current activity and executes so called *interrupt handler* routine, which is usually a fragment of a device driver (in this case: the fragment of the network card driver). The interrupt handler usually moves the data to intermediate FIFO buffer called a *backlog*. For other cards, which do not use DMA, the interrupt handler is responsible for transferring the data from the NIC's internal memory to the backlog. The depth of the backlog queue may be tuned - it is usually set to hold a few hundreds of packets (by default: 300).

Once the data is in the system memory, it could be processed by the *protocol stack*, i.e. the data packet may be classified and processed through routines specific to IP protocol, TCP or UDP protocols, etc, which may, for instance, re-assemble the message that needed to be fragmented to transmit in form of packets, request the re-transmission of the packet, detect that the packet is corrupted, etc. Typically, these routines are quite time-consuming, and should not be executed from within the interrupt handler: note that when the interrupt handler is running, prosaically no other interrupts (e.g. caused by keyboard, mouse, other network cards in the system, etc) can be processed: the system would loose the responsivness under heavy load. Too high rate of the interrupts, and too long interrupt handler may lead to the situation of *livelock*, where the system seems to be "hung", because it spends all the CPU resources on handling the interrupts. There exist techniques for *interrupt moderation*, and will be described below. As described above, the interrupt handler should not execute the protocol-stack routines immediately. Instead, it uses the mechanisms of so called *bottom halves*, or *soft interrupts*. It raises the flag in the system, that the protocol stack routines should be executed as fast as possible, if there is no other time-critical actions to do - it *marks the network bottom half for execution*, then ends, which means that the processor resumes the execution of its original actions.

Till this point, we may identify the following queues:

- the memory in the network card, to which the packets are stored on reception; the system memory to which the data is copied by DMA is not exactly a buffer: it is usually a *memory mapped* region for the driver, with well-controlled bounds.
- the backlog queue, in which the packets are copied by the interrupt handler; the packets are waiting in this queue until the *network bottom half* is executed.

Every time the system scheduler is called (i.e. when the system timer ticks to change the current job, or whenever an interrupt handler finishes its job), it checks the list of bottom halves to find the ones that are marked, and executes the requested ones. Note, that the execution of a bottom half has lower priority than the interrupts, yet it still preempts the execution of user-level programs (tasks); that is why it is also called a *soft interrupt*. The soft interrupt may be pre-empted by any hardware interrupt, but it will be resumed (and ultimately completed) after the interrupt handler is finished.

The network bottom half executes the routines of the protocol stack, as described above. Ultimately, it copies the resulting data (re-assembled and ordered packets, etc) into one more queue, called socket buffer. A socket buffer is the memory area, from which the data is accessible by the user-level applications. Each new socket being open creates a new socket buffer for read and write operations. The size of this buffer may be set by programmer - by default it is 64 kilobytes, which corresponds to around 40 packets. Once the data is copied to the socket buffer, the task that was put asleep in the call to `recv()` function is marked for execution. When the scheduler finally wakes it up (resumes its execution) the data from the socket buffer is moved to the memory area specified by the programmer.

Let us now identify the places where a packet loss may occur, and describe congestion-prevention methods.

The first queue where a congestion may occur is the internal buffer of the NIC. The Ethernet technology specifies the technique that can be used to prevent the congestion here. When the NIC's buffers are becoming full, a special type of packet: Flow Control packet is sent to remote end, to specify that the transmission of further packets should be suspended. Ethernet Flow Control is usually a configurable option for the NIC drivers and for network switches, and it is usually implemented by the NIC hardware. We will not describe it further here. However we want to note, that with the speed of the modern PCs, and the high-performance design of interrupt handlers, usually there is no need for Ethernet Flow Control to protect the NIC's buffer - the data is transferred to the system main memory at sufficient rate and the Ethernet Flow Control frames are practically never observed on the lines connected to PCs (the situation is however different on the *up-link* lines connecting cascaded network switches).

The second queue is the backlog queue. Usually its depth is sufficient and it is not overloaded. At this point, however, let us discuss in a bit more detail the situation of the *livelock*, mentioned already above. The livelock condition may be created when the rate of the interrupts being raised is so high, that it consumes the majority of CPU resources. The *livelock* situation started to be realistic again, when the Gigabit Ethernet and 10 Gigabit Ethernet NICs started to be available. With the throughput of the Gigabit Ethernet, short packets sent back-to-back on the line will arrive with less than 1 μs delay. If the standard processing mechanisms, as described above, were used, every received packet would raise an interrupt, which would lead to a critical situation where 1 milion interupts per second would need to be handled. With typical interrupt service times of at least a few microseconds this would certainly lead to a *livelock* situation. To alleviate the possible problems, interrupt moderation techniques were implemented in the drivers, and in the kernel infrastructure. The main concept is to service multiple packets within one execution of interrupt handler (the costs of setting up the execution of the interrupt handler are relatively high, hence the possible gain in performance). One of the solutions, implemented in the network card drivers, is so called *interrupt coalescence*. Using this solution, the system configurator can tune the maximum rate of interrupts being raised. Note however, that limiting the rate significantly will lead to increased latencies of packets, hence this technique needs to be fine-tuned to the actual network traffic pattern.

The second technique called dynamic interrupt coalescence is a universal infrastructure of the Linux kernel, called *NAPI* (New API for network drivers), from which the authors of network driver may take advantage. It dynamically tracks the rate between the interrupts and monitors the backlog queue, then dynamically controlls the interrupt throttling parameters, depending on actual load of the system. This new feature is however not yet widely used.

Let us now turn back to the third queue in the receive process: the *socket buffer*. This is in fact the only place where the packets may be lost, once they have been received correctly by the NIC. The loss may in some cases be eliminated by increasing the size of these buffers, yet it is not a universal method. Let us look at the problem in a bit more detail. The queue overrun condition happens if the data is appended to the queue faster than it can be absorbed from the queue, and this congestion condition persists for a long time. It is straightforward to realise that by increasing the length of the queue in a significant way, one may prevent packet loss in situations when the congestion is more and more prolonged. However, the culprit of the queue congestion is actually not the inceased incoming traffic rate, but rather lousy application that cannot cope with receiving and processing of incoming data! Hence, usually an improved design of the application may resolve the problem of

lost packets. Usually, the application is designed in such a way, that the data is received then processed in a loop. If the processing step is too time-consuming, that leads to the situation where the data is not moved away fast enough from the socket buffer. There are in general two solutions to the problem: one is the implementation of some kind of flow control mechanism, the other is dedicating a high-priority thread to deal with communication, and separate the processing in another thread. The former solution is in particular what the TCP/IP protocol does by means of *sliding windows* mechanisms: it tracks the free space of the socket buffer and transmits this information to the remote side, to control the amount of information that could be sent.

Send process

Let us now complete the picture of Linux network communication by a short description of network *send* process, and the description of the access control mechanisms there. Typically the send operation consumes slightly less CPU resources, and is conceptually simpler. When user's application calls the `send()` function, the data, which is to be sent, is copied to the associated send socket buffer, and the function is put asleep, as in case of the `recv()` discussed above. The kernel then copies the data from the socket buffer to the NIC's send buffer and tells the NIC to start the transmission of the frame. Once the data is moved from away the socket buffer, the process which called is woken up. In case the NIC was stopped by reception of the Ethernet Flow Control frame, the flow control information is propagated back: the process of copying from the socket buffer will be suspended, and ultimately, the `send()` function will be suspended (or hold in the sleep state by the scheduler), until the send is possible. Having said so we stress that packet cannot be lost inside a PC due to any buffer overrun.

7.2 Measurement methods and tools

To test the performance and prepare the parameterisation of the network routines we performed network communication tests, following the general approach used already in the early TCP/IP protocols tests, documented in [79]. To minimise the complexity and the impact of external influences, we prepared a *testbed* setup made of two identical PCs, with direct network connection between them (i.e. without any switches between). The two PCs were running the same version of (usually customised) Linux kernel. Both machines were equipped with Intel Xeon processors running at 2.2 GHz clock speed and the `Intel e1000 Gigabit Ethernet` network adaptor (although physically the machines were equipped with two processors, with `HyperThreading`, they were booted with non-SMP kernels, hence they behaved as if there was a single processor in the system).

With respect to the tests using Fast Ethernet network, as performed previously, the time scales for events shank by a factor of ten, and single microsecond time resolution was required. That is why we gave up the request-response tests and concentrated on the data streaming tests.

We prepared a testing application, which was transmitting a stream of data packets with predefined rate between two: the client (the machine that sent the data) and the server (the machine which received the data). The testing application was opening the socket (for sending - on the client machine, for receiving - on the server) for one of the specified protocols: TCP, UDP or Raw Ethernet. A predefined number of datagrams, the length of which was also defined for a setup case, was then transmitted from the client to the server. Each datagram contained a *sequence number*, hence at the end of the run, it was possible to determine the fraction of

packets that were lost (not received by the server application.

Testing program

The client application was executing a loop with `send()` function and a construct that implemented a controllable delay. Due to the fact that standard Linux kernel is not designed to be run in the real-time system regime, the scheduling of the subsequent send operation, inter-spaced with controllable delay was a particularly challenging task, that required the understanding of the internals of the operating system. Even though the `sleep()`, `usleep()` and `nanosleep()` functions are available in Linux to introduce a delay in the execution, the actual time of the sleep is a "best effort" approximation, and cannot be guaranteed: it depends on the activity of the system (load, number of interrupts), and the precision better that a standard kernel clock *tick*, which is 1 or 10 milliseconds cannot be guaranteed. In our tests, however, we required the delays to be controlled at a precision of a fraction of a microsecond! The only possible implementation was to implement the delay as a tight "`for`" loop, with empty (or trivially simple) body, and the number of loop executions controlling the delay. By assigning the *real-time*-like scheduling for the process, raising its priority to maximum, we had a good chance that the client program was not preempted by another process, and run without glitches, with even distribution of the delay time. Due to such implementation of the delay function, the actual rate at which the packets were sent was not known a-priori when the test was run. It was calculated for every run on the server, by dividing the number of received datagrams by the time for execution of the run.

The server application was executing a minimal, *tight loop* built of the `recv()` function, and the checking of the sequence number of received datagram, so that the end of the test could be determined (by reception of the datagram with a sequence number larger that predefined maximal one). The server application also measured the time that elapsed from the reception of the first packet, till the reception of the last packet.

To derive the information about the absolute timing of a single operation, from the rate, we also needed to know what the average CPU utilization was during the run. Although the statistics for 1, 5 and 15-minutes average load are available in the system, their precision and granularity were not sufficient. Hence, we developed our own CPU load measurement method, based on the `cyclesoak` program [81]. The concept of this method is to run a low-priority thread, which executes a simple loop with a counter. By comparing the number of loop executions per second on a system under test, with the number of executions on an idle system one estimates the system load. The method turned out to be reliable, yet the drift of results up to few per-cent could be observed.

We executed the test on the machines with various settings for system kernel parameters controlling the network subsystem operation, and various settings for the network card driver. The most important variable being tested was the interrupt coalescence setting.

The tests performed using the procedure and the program described above established the upper limit for the performance of the network communication on the available hardware. The aim of them was to characterise the behaviour of the Linux networking routine and to parameterise it for the purpose of the modelling, that is why the contents of the testing routine was practically empty. In particular, the data being sent or received did not any memory management, etc. Therefore, the second stage of testing was performed, aimed at the realistic tests of the Message Passing libraries of the Data Collection Software. The general methodology for the test remained the same - the only change was the replacement of the functionality provided by the system call functions (`recv()`, `send()`), with their respective coun-

terparts from the Message Passing library. The associated memory management routines were also incorporated, so as to get a fully functional, trivial streaming application that fully used the Data Collection Software infrastructure.

Profiling methods and tools

A general method for gathering the execution timing information (called also: *profiling*) is a standard method of identifying the performance bottlenecks. Typically, the application code is instrumented with statements which leave *timestamped* traces of run, e.g. prints the actual time and a short description of the routine being executed. Later, the timestamped printout may be analysed to determine the time needed to execute certain parts of the code. This particular method was already used to measure the values for parameters of the early model. It is however not possible to *timestamp* the activity of the Linux kernel this way! In our study, however, we could use the `OProfile` tool [82, 83] , which was invaluable to understand where the time is spent in the system. `OProfile` produces the profiling trace of the running system, including all activities of the user-space program and the operating system kernel. During the traced run, a special, very low-overhead interrupt is raised regularly by CPU itself, and the contents of the *Instruction Pointer* (which is the address of the currently executed instruction) is recorded. Once the run is finished, the list of the addresses is resolved to the names of the actual procedures, and the statistics containing the name of the function, the number of time it was recorded and the relative percentage with respect to total number of recorded numbers are given. What is also important, the time when the CPU was idle can also be determined.

The `OProfile` tool takes advantage of the CPU internal performance counters, in order to generate the interrupt in regular intervals. In fact, these counters, which are available on the majority of processor types for some time now, may deliver very fine-grain timing information: one of the counters is for instance incremented with every clock tick! Because the operation of reading the value of this counter may be performed really fast (in few CPU clock ticks), this gives a timing precision at the level of better than a nanosecond on the CPUs working with gigahertz-range clock. Unfortunately, due to the always changing thermal drift of the clock, this method is not perfectly suitable for absolute measurements of short times: the CPU clock is far from delivering the constant clock signal with the precision of a fraction of the nanosecond. The other problem appears of multi-CPU machines, where the processors are usually not boot at the same time, but rather in a sequence, hence their counters are not synchronised in any way, and the actual CPU, on which the program is executed needs to be traced. Regardless of these problems, the profiling using the CPU-performance counters is implemented in the Data Collection Software, and may optionally be activated. The most important advantage of this method is its very small overhead, when compared to gettimeofday() function, which is a system call and requires very time-consuming (the overhead may hence be of order of a millisecond!)

7.3 Results of tests and measurements

The results of the Linux network performance measurements and studies were partially documented in [84]. Here, let us present the most important results, which are relevant to the modeling and the performance of the ATLAS Data Collection system.

TCP connection-demultiplexing mechanisms

Let us firstly discuss the early studies of the TCP/IP protocol performance. The TCP/IP is a point-to-point, connection-based protocol. In the systems where the number of communicating elements become large (e.g. the full-scale ATLAS Dataflow System), problems with scalability of connection (socket) demultiplexing may appear. As an example, let us focus on a ROS node, from which data is retrieved by a few hundreds of L2PUs and a hundred of SFIs. The ROS application needs to track the state of all connections and be able to recognise the situation when a new request comes on one of the connections. Constant polling all the connections in a loop, and trying to read the data is obviously the least optimal, brute-force solution. The operating system provides a few standard mechanism dedicated to this task, the most popular of which is the `select()` (or its variation: `poll()`) system call. Although the select() method is very comfortable from the point of view of the application programmer, it is known to have scalability problems. Hence, if TCP/IP were to be used in the full-scale ATLAS Dataflow system, another method, based on the system mechanism of signals (`SIGIO`), should be used. The `SIGIO`-based method is more difficult to implement and debug. In the figure 7.1 we illustrate the scalability problem of `select()`, and demonstrate that signal-based method scales well with increasing number of connections.

The problem of the scalability of the `select()` function seems to become serious only when the number of connections exceeds a few tens; one should therefore have in mind that the performance problems may not be visible on scaled-down prototypes of the system, but will become serious for the full-scale system.

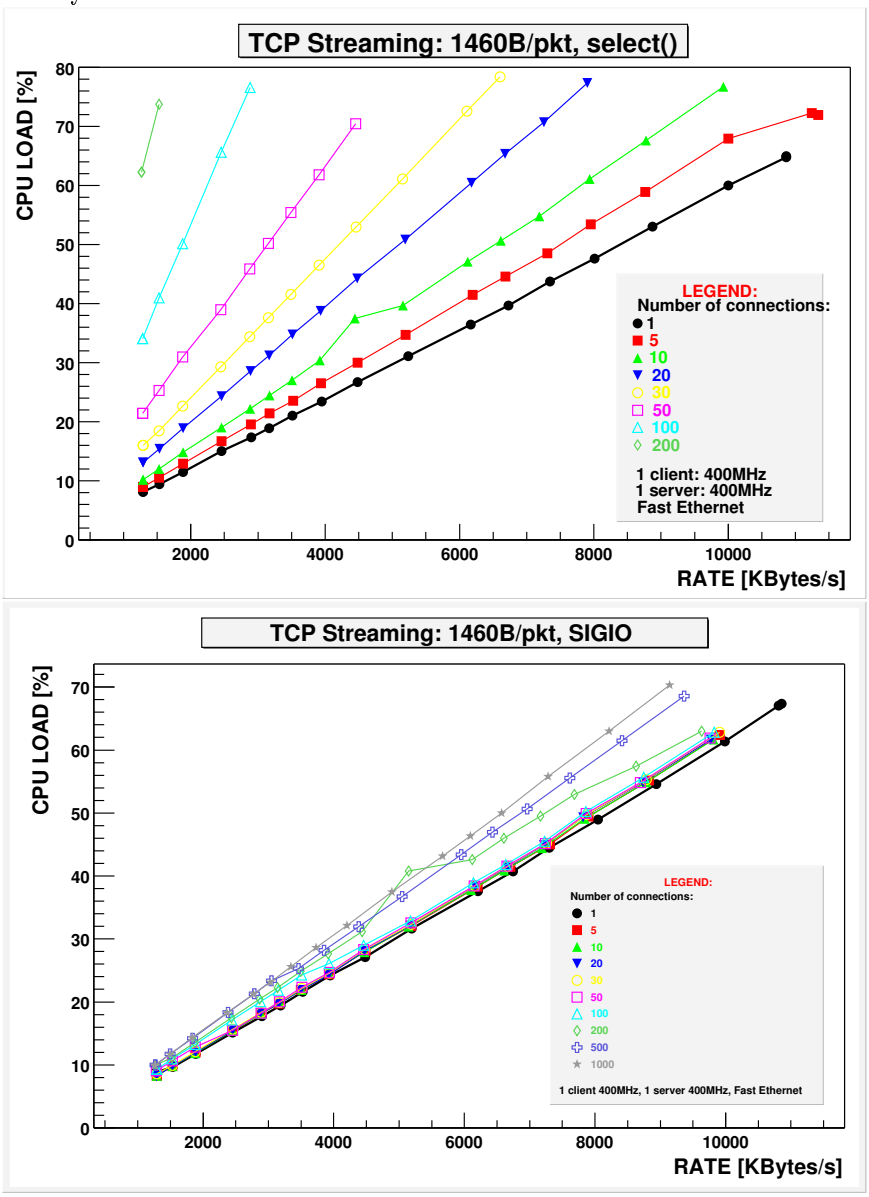
Socket buffer size for Gigabit Ethernet connection

The first important conclusion from the tests we performed on the Gigabit Ethernet testbed was that for the streaming tests, where 1500-byte data packets are sent back-to-back every 12 microseconds, the default size of the socket receive buffers, 64 kilobytes, is insufficient. Raising the value of this tunable system parameter to, say, 1 megabyte already resolves the problem of lost packets in the test cases with big packet streaming tests. Later, this information appeared also in the documentation of the driver for the card.

Secondly, to achieve line-speed data transfer (i.e. to approach the maximum speed available on the Gigabit Ethernet media: around 120 megabytes per second), interrupt moderation techniques are required. The need for interrupt moderation was even more dramatic for the cases of streaming of small datagrams. We determined, that it is quite hard to achieve lossless transmission with packet of small size (64 bytes). We also determined, that the default settings of the interrupt coalescence parameter of the driver depends on actual chipset of the NIC, hence a lot of care was needed to verify that network cards always have the correct settings, when the tests were to be repeated on a testbed made of another set of the machines, which seemed to be identical.

The above discussion of conditions for packet-loss-free transmission obviously refer to the tests using the UDP/IP and Raw Socket protocols. In the case of TCP/IP , the logic built-in the protocol takes care of the packet flow control and congestion avoidance. The tuning of the interrupt coalescence and the size of the buffer was however crucial to achieve top speed in the transatlantic PC-to-PC transfers over 10 Gigabit Ethernet medium, in scope of the tests of the WAN PHY we performed [85].

Figure 7.1: Problem with TCP/IP socket demultiplexing scalability using `select()`, resolved by `SIGIO`



Interrupt mitigation techniques

The studies of interrupt mitigation techniques, presented below, have been performed between 2002 and Spring 2003, and they do not reflect the *state-of-the-art* of studies available nowadays. Since that time, two important sources of informations became available. Since September 2003, the Intel Application Note AP-450 [86] is available; it describes the use of the interrupt coalescence in the `Intel e1000` network cards. Secondly, the article [87], presented in April 2004, discusses the impact of the interrupt coalescence on various aspects of network performance. We will however discuss our own results, because of their significant impact on the way the at2sim model was being developed.

In the discussion below we will mainly concentrate on the interrupt coalescence for receive process. We would point out, however, that the interrupt coalescence for the send process also brings significant improvements to the performance of the data send process. Its impact is visible in particular for the streaming tests, where it allowed us to reach packet rates exceeding 230 kHz (64-byte packets) when the transmit interrupt coalescence was active (with default settings of the driver), while hardly half of this rate could have been achieved when transmit interrupt coalescence was deactivated! We will also point out that for request-response type of traffic the transmit interrupt coalescence being active does not affect the latency of the packets.

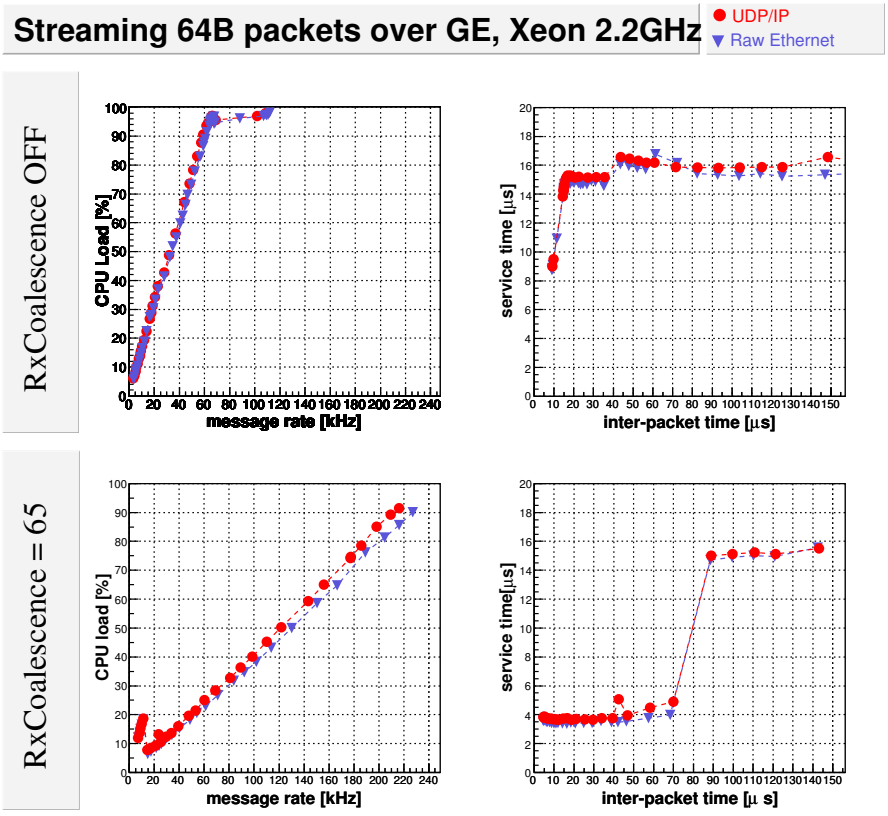
The results of streaming tests for setups with interrupt coalescence active and inactive are presented in Figure 7.2 (for 64-byte packets) and Figure 7.3 (for 1450-byte packets), for UDP/IP and Raw Ethernet protocol.

The plots on the left side of the figures present the direct results of measurement, i.e. the system load on the server, which was receiving the stream of messages transmitted at specific rate. The plots on the right side of the figures present the same results transformed to more useful parameters space: the dependence of average CPU time needed to receive a single message in function of the average inter-packet time (gap) is plotted.

The top-pairs of the plots in both figures present the results for test with interrupt coalescence switched off. The CPU load depends (approximately) linearly on the message rate, and the average processing time does not depend on the rate (and inter-packet time). The end-point of the characteristics that refer to the highest rates (and shortest inter-packet time) refers to the conditions where the measurement behaved in an unstable way, due to CPU saturation. For the message rates higher than the ones plotted we observed packet loss, hence these point seem to be an unstable "transition" point. We do not try to explain the actual behaviour in this area. Instead, we conclude that for streaming at stable condition the average CPU time needed to receive a message is around $15.5\mu s$ for 64-byte long packets and 17 μs for 1450-byte long packets.

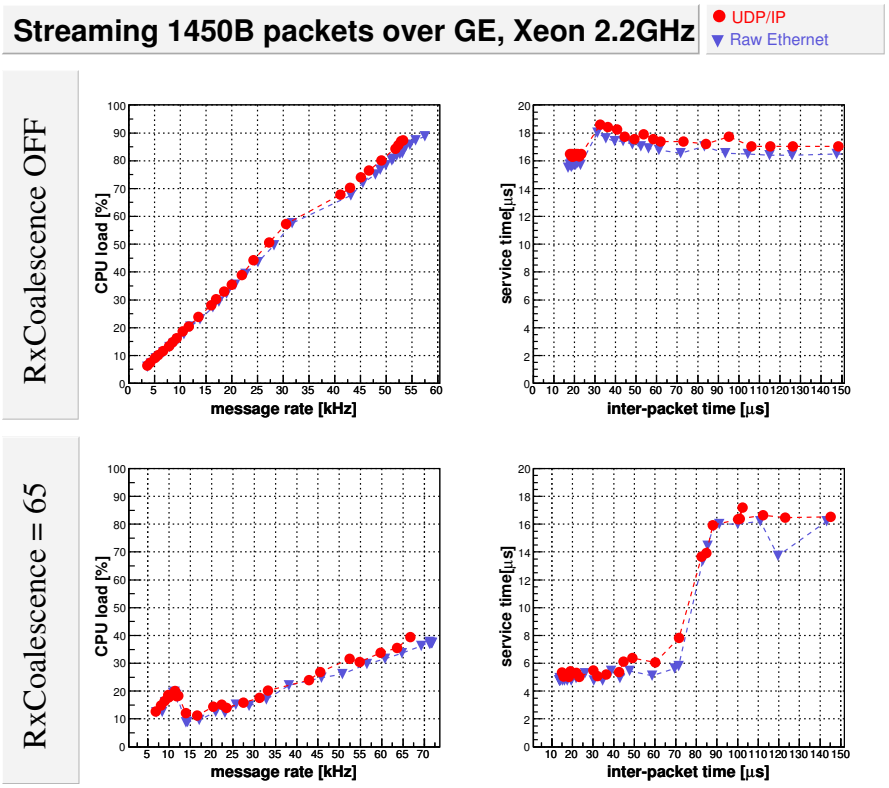
The bottom-pairs of the plots present the results for tests where receive interrupt was set to 66.5 μs. The action of the interrupt coalescence is clearly visible in the right-side plot. On the left-side plot, the effect is also visible as a "bump" at the message rate of around 15 kHz (which corresponds to the 66.5 μs interrupt coalescence time. For small message rate (inter-packet gap larger than around 90 μs), the effects of the interrupt coalescence is not visible, and the average time to receive a message is again around 16 μs. The effects starts to be visible for larger rates: in the bottom-right plots a quite steep, *step-like* transition area followed by a flat, plateau characteristics for smaller inter-packet times is clearly visible. In this region, the reduced average service time is due to multiple messages being serviced by a single interrupt, hence the height of the step reflects to the time spent in the interrupt handler: around 10.5 μs. The height of the left plateau on bottom-right plots, around 4.5 μs, may be interpreted as being spent in the protocol stack. These

Figure 7.2: A typical characteristics with Interrupt Coalescence activated.



The measurement performed on 1*Xeon 2.2 GHz, RxCoalescence=65, streaming of 64-byte packets over UDP/IP and RawSock protocols

Figure 7.3: A typical characteristics with Interrupt Coalescence activated.



The measurement performed on 1*Xeon 2.2 GHz, RxCoalescence=65, streaming of 1450-byte packets over UDP/IP and RawSock protocols

values are used in the message-passing parameterisation of the at2sim model.

Comparing the left-top and left-bottom plots on Figure 7.2 one could clearly see the advantages brought by the interrupt coalescence. Firstly, the maximum message rate that can be withstood without packet loss in the test where interrupt coalescence was used (230 kHz) is at least two times higher than the one that could be withstood without interrupt coalescence (around 115 kHz, if the area of unstable behaviour is taken, 70 kHz if the linear part of characteristics is only taken). In the left-bottom plot, the slope of the very beginning of the characteristic is equal to the slope of the left-top characteristics.

The second advantage is visible on the plots in the Figure 7.3 the maximum throughput (approaching the line speed), which corresponds to the message rate of around 70 kHz, may only be reached for the setup with interrupt coalescence; the measured CPU load was around 40% in this case. In the setup where interrupt coalescence was not active, sustained stable (packet-loss free) transmission with maximum throughput could not be achieved due to CPU saturation.

The effects of the interrupt coalescence on the performance in streaming tests for UDP/IP and Ethernet traffic, as observed above are clearly beneficial. It should ,however, be realises that any kind of static interrupt moderation technique affects the latency of the transmission. In the Figure 7.4 we present the results of request-response tests, performed on setups with various settings of the interrupt coalescence timer (the one-way latency, which was calculated as the half of round-trip time is plotted). As expected, interrupt coalescence linearly adds the latency up to a certain point (around 20 μs in this case). This point corresponds to the setting of another parameter of the interrupt coalescence, which is a timer responsible for ensuring that the message latency will not exceed certain value. It is therefore vital to optimise both parameters for request-response type of traffic. Let us note here that the increased latency due to the coalescence may have an influence on the optimal setting of the SFI credits parameter.

To complete the presentation of the test results of the interrupt coalescence tests, in the Figure 7.5 we present the characteristics for tests performed on a two-processor machine. It becomes apparent that the mesage-processing performed by the kernel routines needs some locking and cannot take full advantage of both available processors. In the setup without interrupt coalescence, packets start to be lost at the rates of 130-170 kHz due to the resource saturation, even though the total system load approaches 50%. This clearly indicates that although the Big Kernel Lock (BKL) has already been eliminated from the kernel, still significant amount of locking is present (and probably cannot be easily eliminated).

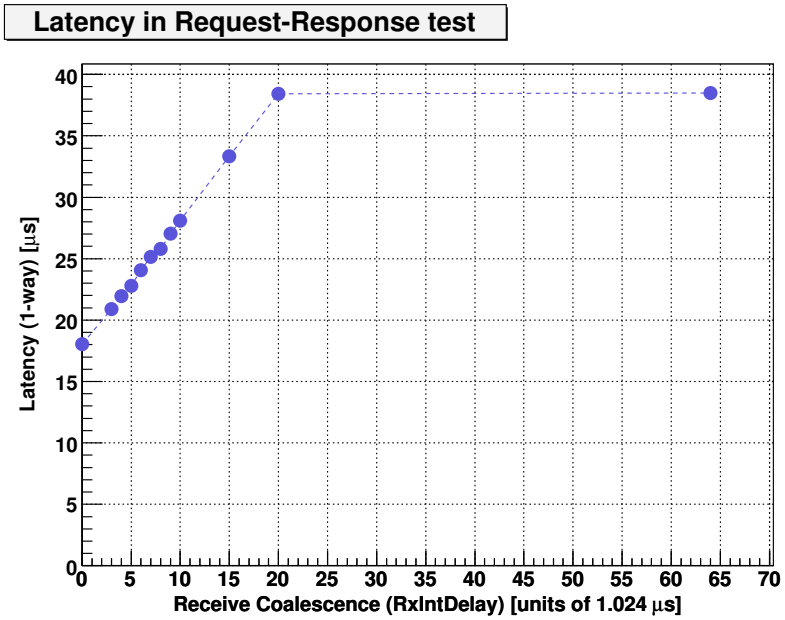
`NAPI`

In the Figure 7.6 we present the results of a streaming test for setups where no interrupt mitigation was used and the results of a setup where the option for the use of the `NAPI` infrastructure for the dynamic interrupt moderation was used.

The impact of the use of the `NAPI` on the performance of the network receiving routines is similar to the one of the interrupt coalescence. We should note that at the time when these tests were performed the `NAPI` infrastructure and its support in the network card driver was in an early (experimental) stage, and we encountered a lot of problems in establishing stable setup for the tests.

Let us also point out that currently the driver for the `Intel e1000` network card has its own option for dynamic interrupt coalescence, hence the advantages related to the use of `NAPI` may be questionable in this case.

Figure 7.4: Coalescence leads to increased latency



The measurement performed on 1*Xeon 2.2 GHz, request-response test using 64-byte packets over UDP/IP; Tx Coalescence has practically no impact in this kind of test.

Figure 7.5: The impact of the interrupt coalescence on system load on double-processor system.

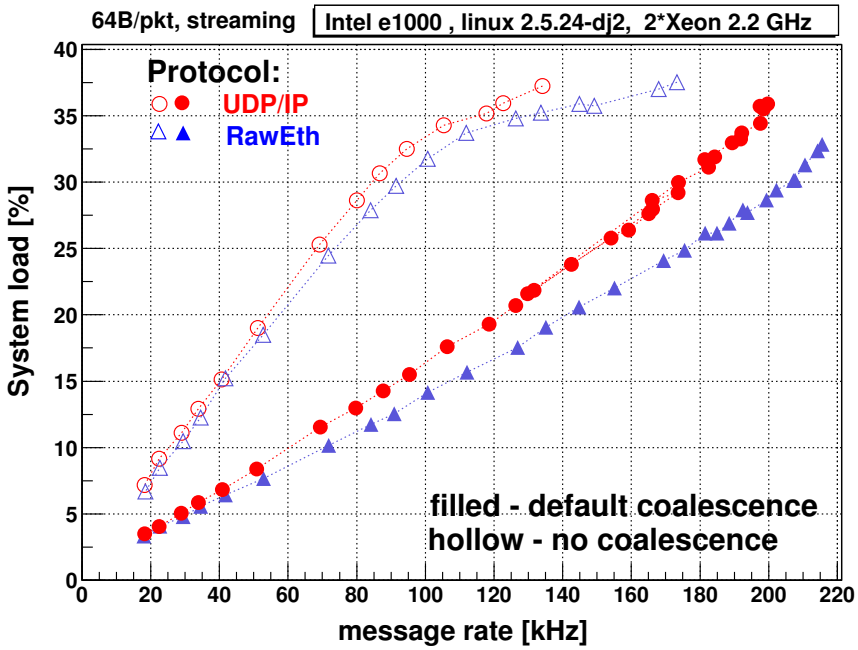
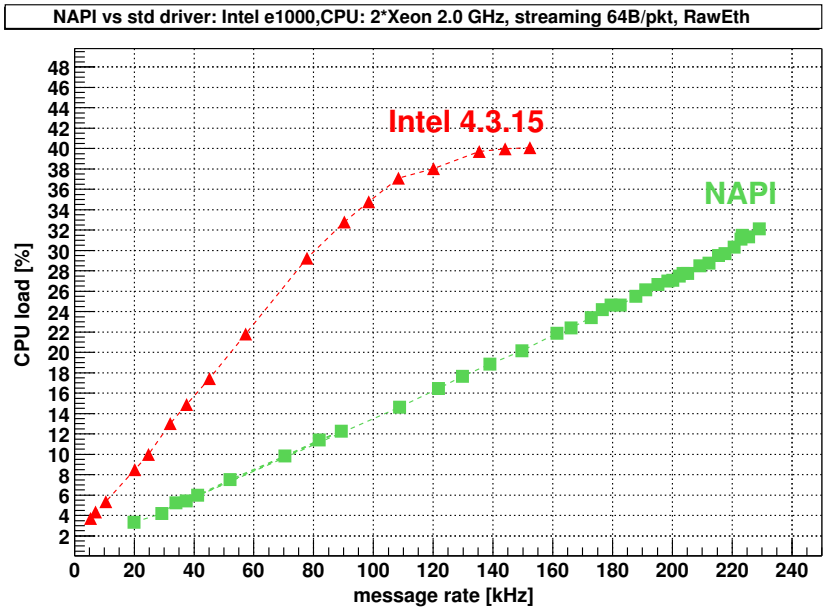


Figure 7.6: The impact of the interrupt coalescence on CPU load



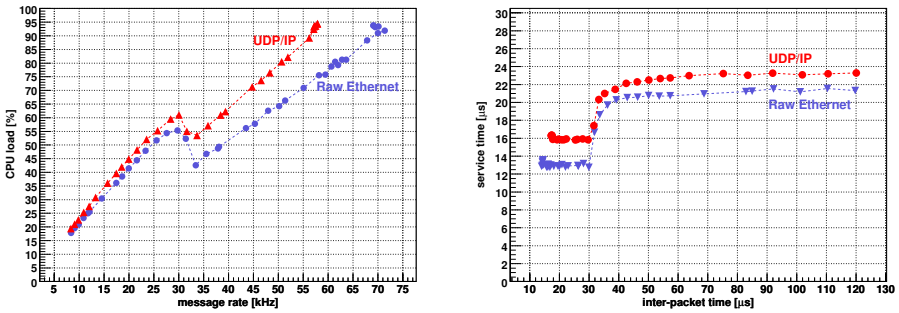
Measurements for Data Collection Software

In the Figure 7.7 we present the results of the streaming tests performed with the Data Collection Software message passing, rather than "raw", lower-level operating system APIs. In this case, the interrupt coalescence timer was set to around 30 μs, which is clearly visible in the characteristics.

From the comparison of the right-side plot in Figure 7.7 and the bottom-right plot of the Figure 7.2 one can determine the CPU overhead introduced by the Data Collection Software Message Passing libraries: the shapes of the characteristics are similar and differ by a constant shift of around 8 μs. Again, we will use the value obtained here in the parameterisation of the at2sim message passing.

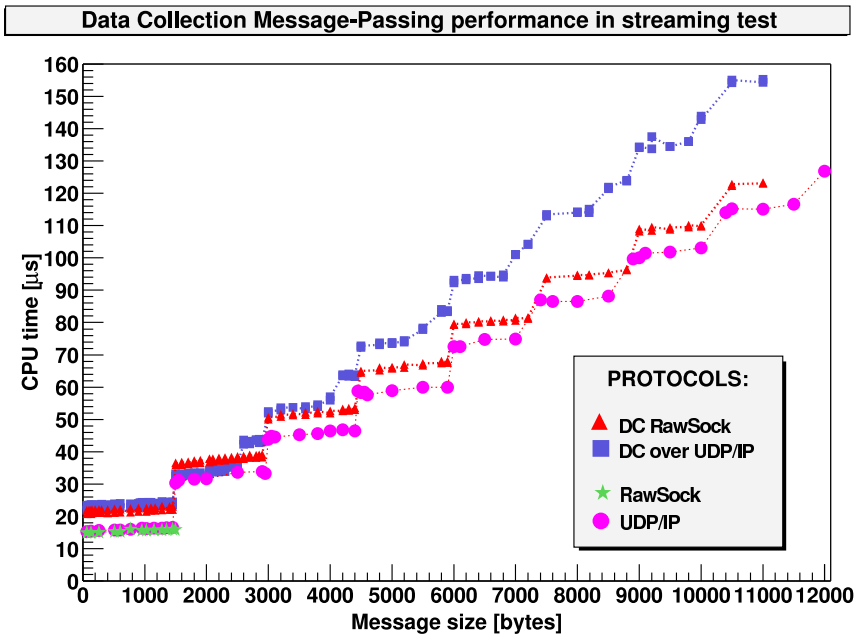
In the Figure 7.8 we present the results of the streaming tests with Data Collection Message Passing, performed to determine the impact of the message size on the time needed to process a message.

Figure 7.7: DC Streaming test results



The measurement performed on 1*Xeon 2.2 GHz, RxCoalescence=30, streaming of 64-byte packets over Data Collection Software UDP/IP and RawSock protocols.

Figure 7.8: The results of tests of the Data Collection message passing. The interrupt coalescence was OFF in this test.



A clearly visible step-like characteristics is caused by the packetization of messages: the steps appear for the messages which size crosses the boundary of packet size (around 1500 bytes). The height of the step, which is around 15 μs, reflects the time needed to receive another packet, as already understood. We also observe a small slope on the flat-like parts of the characteristics. This dependency turns out to have around 1% effect, and it is also taken into account in the at2sim parameterisation. We have also determined that the average time to send a Data Collection message is around 12 μs.

Chapter 8

Ptolemy and Discrete Event simulations

The at2sim model is based on the Ptolemy Classic package, developed at University of Berkeley in scope of The Ptolemy Project [74]. Despite the fact that a new Java-based implementation of Ptolemy II was already made available, it was decided that the Ptolemy Classic, implemented in C++, will be used. This decision was motivated by numerous reasons, such as the experience in the use of Ptolemy Classic, better performance, etc.

The Ptolemy project is dedicated to the research in software-based techniques for embedded systems. The Ptolemy Classic (and Ptolemy II) software packages were created as software infrastructure for experiments with design techniques: they have a form of extendable, complex simulation tools, allowing to perform computer experiments in a variety of computational models (refereed to as Ptolemy domains), such as Discrete Event, Synchronous Dataflow, or Finite State Machine. For our purposes, only the Discrete Event modeling was of importance, hence, from this point on, I will refer to the Ptolemy package as a simulation tool in the Discrete Event domain.

The Ptolemy software consists of the basic infrastructure for modeling, a set of standard model components, and the tools to assemble and execute the simulations from the components. Despite the fact that graphical tools for simulation assembling and execution existed, we decided to choose script-based approach. The reasons were that the graphical tools were difficult to use, and working with projects composed of large number of components was not effective. Another reason was that in the script-based approach, the output of simulation runs could easily be captured and analysed.

The Discrete Event domain of Ptolemy, the components of the simulation (actors) communicate via sequences of events placed in time. An event consists of a value and a timestamp. Actors dispatch events (always scheduled in future) to other actors, and react to the events being sent to them. The simulation progresses along the line of simulation time, with all activities of the actors being triggered by events and happening in the moment defined by events. Internally, every Discrete Event simulator maintains a global queue of pending events sorted by timestamp (a priority queue). The Ptolemy DE domain uses a calendar queue data structure for that purpose, to be able to perform the enqueue and dequeue operations in time that is independent of the number of events in the queue. This fact was particularly important for large simulations, such as the ones that were to be performed by at2sim. Yet another important feature of the Ptolemy DE simulator was the treatment of the events with the same statement: the order in which they were

processed is inferred from the structure of the model (i.e. the analysis of the graph structure of the model for data precedences is performed).

8.1 Simulation architecture and components

In `Ptolemy` the simulation is performed on a system made of components, called *Stars*. The stars communicate with each other by means of events: a star may *send* an event to another star, or it may be notified that an event was sent to it. A star may implement self-triggering mechanism by sending events to itself.

Each star contain one or more *ports*, uni- or bi-directional, by means of which it receives and sends the events to other stars. All the stars in the simulation needs to have their ports connected to the ports of other stars. The connections are always of one-to-one type, but certain ports may have more than one connection. A star may only send events to the stars that are directly connected to it. The simulated system has therefore a form of connected graph, with stars being the nodes and connections being the edges.

The `Ptolemy` package delivers a set of standard types for stars, such as variants of queues, forks, delays, printout/histogramming "sinks", "blackholes", etc. These standard stars are very useful for basic studies, but they are not practical for simulation of systems, which elements have complex behaviour. In the case of the ATLAS Dataflow System simulation, a typical component would be a computing node, or a multiple-port network switch. Despite the fact that the behaviour of such element could be modelled using a combination of these simple stars, it would be impractical to assemble the simulation of the whole system, which consists of hundreds of computing nodes and switches, using these primitive stars: not only the whole system would be extremely complex, but also its performance would be drastically limited by the number of "elementary" events needed to model the internal behaviour of individual components. The reason for this is the time spent in the global scheduling queue of the simulator, which makes the scheduling of events a "costly" operation when compared to the time needed to execute the code of the action performed by the star upon arrival of an event. In the case of the `at2sim` model, where resource-sharing and parallel-execution needs to be taken into account, the simulation of even a single component, by aggregating the primitive stars, would almost certainly be impossible! That is how we come to a conclusion regarding the approach to component definition for large Discrete Event simulations: for the large simulations of systems containing the components with complex behaviour, it is more practical to minimise the number of stars in the simulation. This requires that as much part of the functional model component as possible is implemented internally in the model - the components need to be "heavy-weight".

8.2 `Ptolemy` as simulation environment

To perform a simulation using `Ptolemy`, one needs to assemble the system from components (either the existing ones or custom-made), define all possible connections between the components, define the maximum simulation time and start the execution of the simulation. This task may be performed using graphical tools (for simple simulations) or TCL-based script interpreter. Because of the reasons mentioned above, we concentrated on the use of scripts. The simulation terminates when there is no more events with timestamp that is smaller than the defined simulation time. At the end, the statistics and results are usually being displayed, yet in the script-based approach it is feasible to have the printouts in the code of the actors, so that the events and activities may be traced and the evolution of the

system in time may be reconstructed and analysed.

To start a simulation, some star needs to produce the first event - this is usually a part of initialisation code of the stars. This event, in turn, may trigger subsequent events being produced in the stars, which make the simulation evolve. Usually, however, after a sequence of events the activity settles down, because no more events exists. That is why, in a typical simulation there needs to be "initiator" stars which trigger the events by themselves throughout the time of simulation. The mechanism, through which it is possible, is self-triggering by sending a message (with a timestamp at some time in the future) to itself. Obviously, any number of "initiator" stars may exist in the system.

Chapter 9

The new `at2sim` model

Based on the experience rooted in the initial `at2sim` model, as described in Chapter 6, the studies and tests of the performance of the Linux networking procedures and the new implementation of the ATLAS Data Collection Software, the design and implementation of the new `at2sim` model [88] was started. In this chapter I document the steps I performed to create the new `at2sim` model (from now on: the `at2sim` model), parameterise and validate it; the results obtained from the model (with a common effort of the group involved in the `at2sim` modelling) will also be summarised.

9.1 Model decomposition

The decomposition of the model of whole system to the models of components, and the definition of the models of the components are amongst the most critical tasks in a simulation project, defining the balance between its precision, complexity and execution time.

In the case of the `at2sim` model, the first approach to decomposition would simply reflect the physical elements of the real system: the components could reflect the applications of the ATLAS Dataflow System being run on computers, and the network switches. On the other side, there was a heritage of the first `at2sim` implementation, in which the modeling of the network technology was being focused on. One part of this heritage, the parameterised model of an Ethernet switch, was a well-defined entity reflecting this high-level decomposition. On the other hand, a detailed model of a Ethernet network card and simplistic models of the computers and the applications running on them had to be reconsidered.

One of the most important conclusions from the study presented in Chapter 7 was that the activities related to network communication, performed at the level of the operating system may define the performance of a computer-based element of the ATLAS Dataflow System. These activities cannot easily be decoupled from the operations performed by the network card, due to mechanisms such as flow control or interrupt coalescence. On the other hand, in the previous chapter, it was concluded that the number of "stars" in a `Ptolemy` simulation should be minimised in order to obtain better performance and minimise the complexity. To add to this, the model of the Ethernet switch already had its representation for Ethernet packets implemented and intergated with the model of a network card. Taking these arguments into account, we decided that

- the parameterised models of a generic Ethernet switch should be taken in the existing form

- the model of a generic switch will be used to produce the models of actual switches; these models will be callibrated as a result of already ongoing activity in the ATLAS T/DAQ group
- new models of applications will be developed, to meet the requirements of precision
  - the model of the operating system, based on studied presented in Section 7 will be incorporated
  - the low-level model of the Ethernet network card will be incorporated, rather than used as separate star
  - the model of multi-tasking time-sharing execution of multi-threaded ATLAS Dataflow application will be modelled appropriately
  - the models of the network message-passing and operating system will be generic, so the models of any other application could be implemented in future; this generic model may be parameterised and callibrated separately
  - the functional models of the ATLAS Dataflow applications will be based on the code and behaviour of the real applications, in order to provide credible modeling of system behaviour

9.2 Generic model of end-node

The ATLAS Dataflow System is designed to be built using "Commodity Off The Shelf" (COTS) elements: apart from network switches and the elements connecting LVL1 and LVL2 triggers, these will be standard single- or multi-processor computers running Linux operating system. All of them will have standard Ethernet network connectors: one for general-purpose tasks (such as communications and controls) and one or more network cards in Gigabit Ethernet technology to handle the main data flow. The applications of Data Collection Software will be run on all of these nodes - network connectivity will be provided by a common "middleware" layer of the Data Collection Software: the Message Passing libraries (the main task of which is to hide the technical complexity of the network communication, and present a clear communication interface to the Data Collection applications).

I decided to take advantage of this striking commonalities in the design of the `at2sim` model. I developed a generic model of a computer which runs multi-tasking operating system and communicates with other computers using switched Ethernet network. The Data Collection message-passing layer would also be a part of this generic model. The generic model would then be extended (by means of OO inheritance) to implement the specialised models of the components of the ATLAS Dataflow System, such as L2 Processing Units, ReadOut Buffers, etc. This approach has a very important feature originating from the common implementation of key parts of the system: the parameterisation and callibration of the message-passing would need to be performed only once, by performing measurements in a well-controlled, simple set-up. Then, in the first approximation, it would be sufficient to implement only the functional part of the specialised model, without the need to perform the callibration (and in fact: even the whole parameterisation) of the functional models. This is possible, because the performance of all end-nodes is determined mainly by network communication. At later stage, in order to improve the precision, the specialised models could be parameterised and calibrated to introduce the time spent in certain functional execution blocks of their application. The callibration of such parameters could be performed in an easy way, by means

of time-stamping of the code. Again, because the low-level (operating system, networking) routines were already separated-out, such measurements should be easy to perform, and the enhanced models should be easy to callibrate.

Difficulties start to appear, however, when multi-threaded applications are being modelled. Despite the fact, that certain activities being executed in parallel in the real components could be modelled as serialised events (such as interrupts in the network communication), this is usually not the case for the applications. A proper notion of parallel execution needs to modelled for every application that runs parallel threads. In `at2sim`, I created a model of parallel execution, where computing resources are shared between tasks. This approach is particularly important to properly simulate the execution on a multi-processor machine, where certain activities are executed really in parallel. To make the model performant, I decided not to go into much detail in the parameterisation and callibration of the task-switching mechanism of the operating system: proper, detailed implementation, taking into account the time needed to switch the tasks, etc, would be far too "heavy" to be used in the complete model. On the other hand, classical synchronisation mechanisms used in multi-threaded programming (mutexes, locks) need to be implemented.

9.2.1 The model of the task scheduler and CPU resources

One of the key observations that led to the design and implementation of the new `at2sim` was that the discrete event models of components need to use a dynamic system for resource sharing: it is not sufficient to assign static "delay" values to the actions performed in response to certain events. The actual "delay" time should rather be derived, taking into account the state of the component. Despite the fact that the real communication with hardware involves ingredients such as data transfer over (PCI) system bus, we concluded that these operations usually involve CPU to some degree, hence the only resource that should be modeled is the CPU resource. We also observed that in typical applications and setups (as available at that time: 64-bit PCI bus, clocked at 66-133 MHz, Gigabit Ethernet NIC, Intel Xeon CPUs clocked at 2 GHz) usually the CPU is the ingredient that limits the performance, hence the delays due to PCI bus transmission will be included in the ones due to CPU resource consumption.

In a typical multi-tasking operating system, such as Linux, the CPU executes instructions in two contexts: system context and application's (or user's) context. The system context executes the critical operations such as communication with hardware, filesystem operations, and task scheduling (i.e. management of granting the CPU resources to processes and tasks). The applications run in application context, as concurrent processes (tasks) and use *system calls* to request execution of actions in the system context (e.g. an application uses `recv()` system call to receive data from the network). All the activities in the system are prioritised, with all system-related activities having priority over user's application. It is the role of the operating system to schedule the tasks, i.e. to decide when the code of one application should start to be executed, and when it should be *pre-empted*, (i.e. put asleep, suspended) in order to execute the code of another application (process, task) being run in the system concurrently.

In a stationary situation (no hardware communication needed), the task scheduling is governed by the system clock, which drives the scheduler; the tasks are granted the CPU execution timeslot of (typically) 10 milliseconds, in a prioritised round-robin order. However, if a task performs a system call, which cannot be completed immediately (i.e. the situation when data has not yet come over the network, or the disk is not yet ready to return next fragment of the file), the process is put asleep (the process blocks in the system call), and another process is scheduled. The process that was blocked will only be woken up (and resume its execution),

once the reason that blocked it disappeared (e.g. the data arrived from the network or the disk), and the process is selected by the scheduler for execution.

The "stationary" situation, such as described above is, however, not a realistic one: there is always some activity in the system that involves the interaction with hardware (even if there is no network communication and no files are used, the disk is usually used as a *virtual memory* (paging space), hence usually some Input/Output activity always happen). Communicating with hardware is usually a critical activity, which often requires to be completed, without being interrupted, within specified period of time. In fact, the Input/Output operations are typical examples of asynchronous (or event-driven) actions. Typically, the hardware needs to signal the system about the action that needs to be performed on it; in order to maximise the responsiveness of the system, this kind of action requested by hardware needs to be executed as soon as possible, with very high priority. Such signalling mechanism is usually referred to as interrupts.

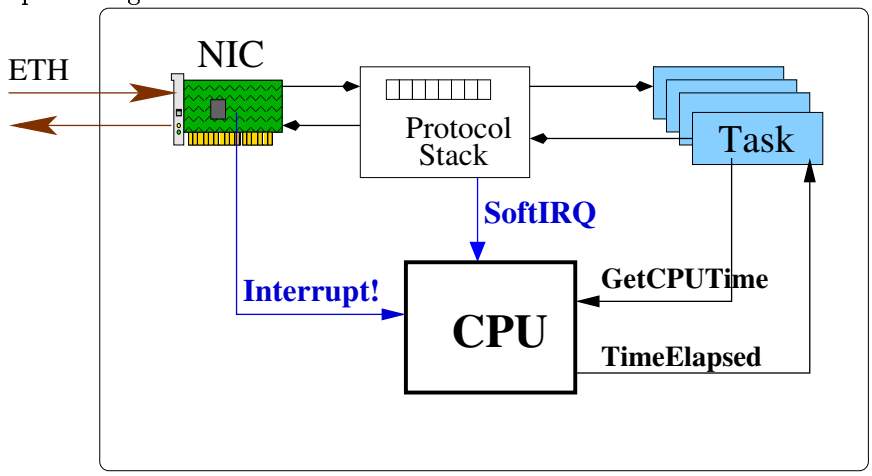
The interrupt mechanism works in the following way: when a hardware elements needs to inform the operating system about its need to be serviced (here, we will use an example of a network card, which received a data packet), they send the interrupt signal to the processor (this may happen directly, or indirectly, by means of e.g. programmable interrupt controllers); the interrupt is indeed an electrical signal sent to the CPU. Upon the reception of the interrupt, the processor immediately abandons the currently executed action, saves its state (usually: to the system stack) and starts the execution of special routine called an interrupt handler. The first command of the handler is usually the command to block further interrupts. Once the activity (which usually involves low-level communication with the hardware that raised the interrupt) of the interrupt handlers comes to an end, the interrupts are unblocked, and the processors restores its state information (that it stored on the system stack when the interrupt came) and resumes the operation of the previous activity.

For the purpose of at2sim, we decided to implement the model of CPU resource (i.e. which may involve multiple processors), shared by a (dynamic) list of tasks, with preemption caused by interrupts. Such design allows us not only to implement the desired parallelism features of the models of components of the ATLAS Data Collection applications, but also properly model the mechanisms and CPU resource consumption due to network communication, taking into account multiple-stage processing and interrupt-throttling mechanisms (such as interrupt coalescence). The model of sharing the CPU resources between network interrupts and user tasks is presented in Figure 9.1.

Each component model (a `Ptolemy` *star*), is a subclass of a `DEApplication` star (which in turn is a subclass of `DEStar` class) represents a single computer, with CPU resources and one or more network cards. Every `DEApplication` contains its own *CPU resource* object (with configurable number of CPUs), which model the operating system scheduler, and the interrupt mechanism handling. The application need to be composed of one or more *Tasks* (objects of a class derived from the `Task` class), which represents the functional models of user's applications. The task may ask for CPU execution time, and they are notified when the scheduled time elapsed. The actual time when this happens is influenced by the activity of other tasksand the network activity involving this `DEApplication`. A task may also request sending a message over one of the network connections to another `DEApplication`, or get notified when a message directed to it arrives (a model of blocking read). Again, the modelled delay of the message, introduced by the operating system and network card depends on the state and the activity of the `DEApplication`; it is calculated and modified dynamically. In the following subsections we will provide more details on the network communication model.

To implement a new model of an application it is sufficient to implement a model

Figure 9.1: Model of sharing CPU resource between network interrupts and requests from processing tasks



for one (or more) task, then instantiate the `DEApplication`-derived class and assign the instance of the new class derived from the `Task` class with the `DEApplication` instance. A few methods of the `Task` abstract base class needs to be implemented for that. The list of methods that needs to be implemented, and the methods that may be used in the implementation of the model of the task is presented in Figure 9.2.

9.2.2 Model of a network card (NIC)

The model of a network card was integrated with the generic model of the application: it, however, exists as a separate class, in order to make it possible to create the models of applications with multiple network cards. The model of the NIC implements the behaviour of an Ethernet card, taking into account Ethernet Flow Control, the input and output queues and the backlog queue, proper timing (with inter-packet gap), VLANs, congestion notifications and interrupt coalescence. The NIC class also takes care of proper translation of the high-level messages, as used in the Data Collection software to Ethernet packets, taking into account Data Collection message format, communication protocol formats (TCP, UDP, IP headers), and packetization. It also wraps the messages in `Amessage` format into a *packet* format that is used by the model of the switch. Finally, it plays an important role at initialisation of the switch routing tables: it automatically sends the *MARP* messages, to make itself known to the switches.

The model of a NIC integrates automatically with the model of CPU resources of `DEApplication`: the NIC generates requests for interrupts, which have impact on the modelled execution of tasks. It uses queues to exchange the incoming and outgoing messages with the model of the networking stack.

9.2.3 Model of network protocol stack

The processing of data through the network protocol stack in the real operating system is a multi-stage process, described in the Chapter 7 above. In the `at2sim` we model these activities for receive process only, using a simple parameterisation of time needed to process one packet through the network protocol stack. As presented presented in Figure 9.1, the processing time is requested from CPU resources, using the model of interrupt mechanism. The applications' socket buffers are also

Figure 9.2: The most important methods of the `Task` class.

Method	Description
`Start()`	executed when simulation is started
`GetCPUTime(int request_id, double time)`	requests CPU time for processing
`TimeElapsed(int request_id)`	called when requested CPU time elapsed (action completed notification)
`Send(Amessage *msg, int request)`	sends a message over network
`ProcessAndSend(Amessage *msg, int request, double time)`	do processing then send a message over network
`BlockingRead()`	enter blocking read, data arrives through `NewDataArrived()`
`int NonBlockingRead()`	non-blocking read (as above)
`NewDataArrived(Amessage *msg)`	called when data arrives
`NewDataOnQueue(MessageType msg_type, list<Amessage*> *msgqueue)`	called when data arrives at message queue (alternative mechanism for net input)
`LockReleaseNotification(Lock *l)`	called when lock was released
`AddTimer(int timer_id)`	adds a timer
`SetTimer(int timer_id, dounle time)`	sets the timer
`CancelTimer(int timer_id)`	cancels the timer
`SetTimerAbsolute(int timer_id, double time)`	sets the timer to absolute time
`TimerTimeout (int timer_id)`	called when timer time elapses

defined, hence it is possible to observe packet losses due to input socket buffer overruns in heavy traffic conditions, and to observe the maximum buffer consumption throughout the simulation run, the latter being particularly interesting.

9.2.4 Model of the common Data Collection Message Passing

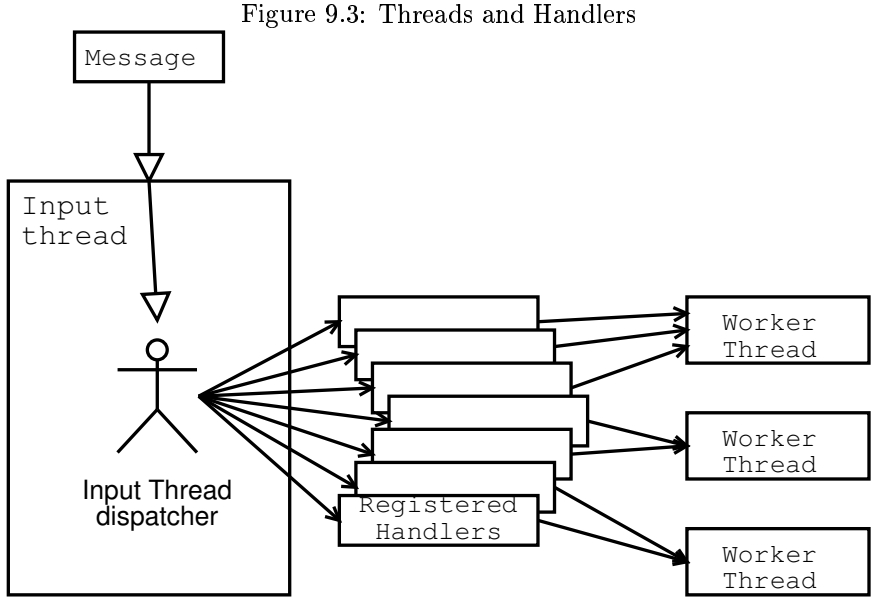
Two models for receiving the data by the application were implemented: one that corresponds to a standard use of primitive network communication functions, and one that corresponds to Data Collection Message Passing mechanism of message queues, message handlers and input thread that handles the incoming messages (the mechanism is presented in Figure 9.3). The message handler mechanism is activated using the `SetAutoReadMode()` method of `DEApplication`. It is used in the model of the L2 Processing Unit.

To simplify the model, we decided to implement all the network input operations as being passed through an *Input Thread* - a thread that really exists in the DC Message Passing, and not in the applications that use primitive network communication operations. The reason for that was to simplify the code of the models of very simple applications - such models may be encapsulated in the body of the `DEApplication`-derived class, without the need to develop a specialised model of a Task - the `InputTask` takes the role of the actual main task of the application, and all the CPU requests due to receiving messages are treated as requested by this thread. It is also easier to analyse the CPU utilisation: the fraction that is due to network communication may easily be separated out.

9.2.5 Parameterisation

The complete set of parameters used for parameterisation of the operating system and applications is documented in [89], therefore we will not repeat it here in full length. Instead, I will summarise the parameters used to parameterise the process of

Figure 9.3: Threads and Handlers



receiving and sending of a (generic) message, the mechanism of which were described in previous subsections.

In the Figure 9.4 we present a diagram with parameters controlling the model of receiving data from the network.

The message reception activity is modelled as a set of queues (in the network card, in the operating system) and tasks that consume CPU resources. Three parameters (all expressed in nanoseconds) are used to controll this process: the `recv_delay` parameter controls the delay related to the network card receiving a network packet; the `recv_int_time` reflects the time spent in the interrupt handler (and, in general, in the operating system routines); the `recv_app_time` parameter reflects the CPU time that the user's application needs to receive a message.

In the Figure 9.5 we present a diagram with parameters controlling the model of sending data to the network. The two parameters controlling the modelling of this process are `send_time`, which reflects the CPU time needed to send a message (on the operating system level), and `send_delay`, that reflects the delay introduced by the network card.

9.3 Models of end-nodes

The parameterisation of the models of the applications (and the parameterisation of the generic models as well) are summarised in the note [89]. The complete list of parameters, with the list of states and state transitions, in the `at2sim` model are available on the web page [88]. In the following subsections a short description of the functional models of the components will be presented.

All models of the end-nodes are based on the generic model of the application and operating system, as described above, and only need to implement the functionality of the modelled component, in a finite-state machine style. The models contain the parameters, the values of which can be measured directly, or obtain by analysis of the results of the tests performed in real hardware and software setups of reduced complexity. The process of extracting the values from the measurements, and introducing them to the models of nodes was refereed to as callibration, and was typically followed by initial validation: confrontation of the measurable results

Figure 9.4: Receive parameters

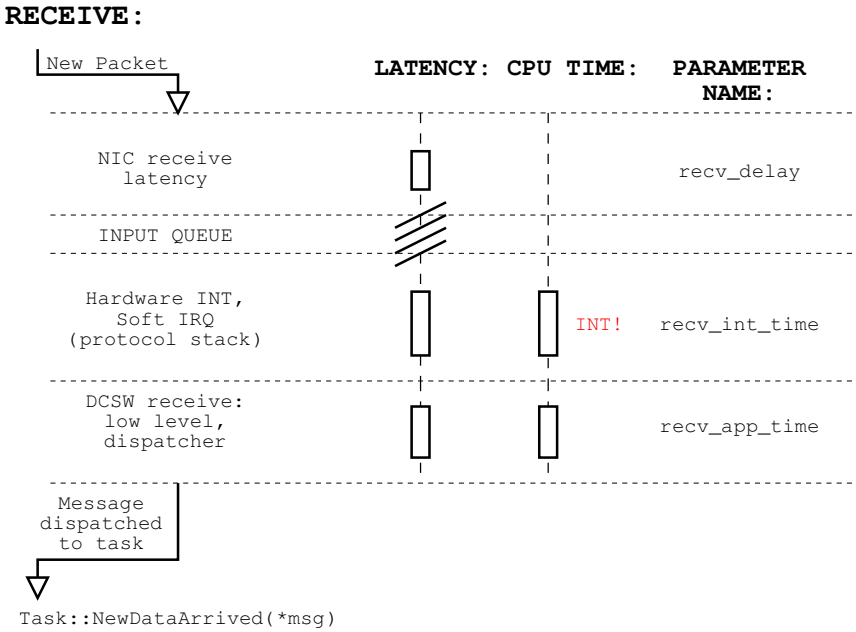
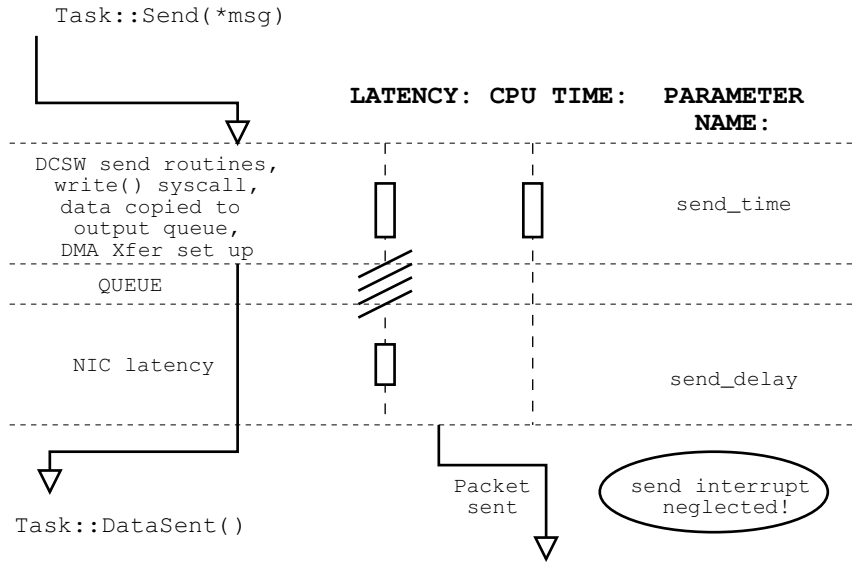


Figure 9.5: Send parameters



of the tests with predictions produced by the model, for these simplified setups.

9.3.1 Supervisor

The model of the L2 Supervisor node (one or more of which may be present in the model of the final system) plays the role of the "event generator". Using the tabularised data for average rates of various types of triggers, it firstly simulates the LVL1 decision being received from the LVL1 trigger (including generation of a RoI areas), then proceeds with simulation of the real L2 Supervisor activities (accumulation of the LVL1 results in the internal queue, assigning them, to the L2 Processing Units, etc). Apart from the part related to the generation of the LVL1 decisions, which was to a high degree re-used from the initial `at2sim` implementation, the model is relatively simply, and uses minimal set of parameters.

9.3.2 L2PU

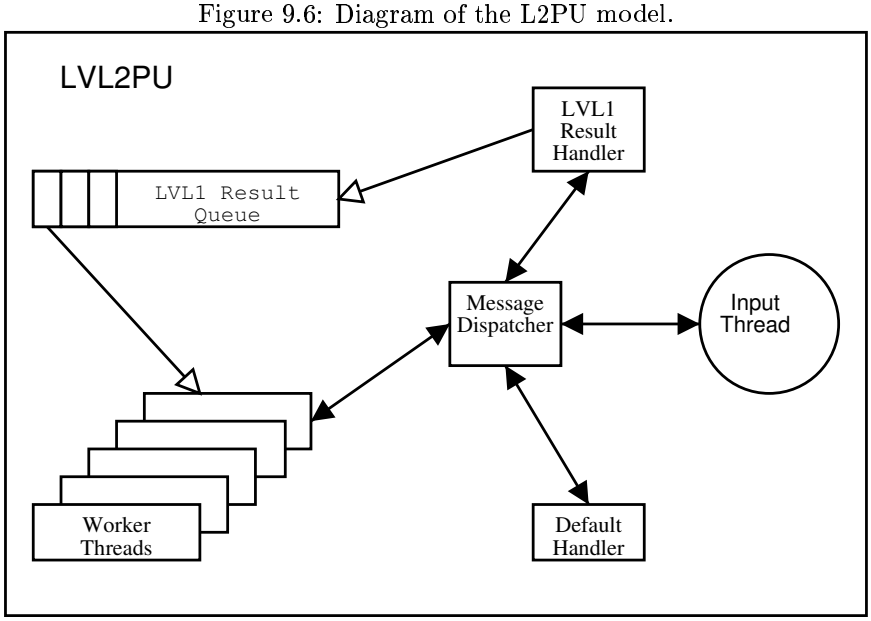
The design of the model of a balanced model of the L2 Processing Unit was amongst the most challenging tasks and drove the design of the generic model of the applications. Each L2PU node needs to execute multi-step (sequential) processing, of undetermined complexity and nature, perform parallel processing of many events at the same time on a multi-processor machine, and use the Data Collection message-passing mechanisms such as dispatchers and message queues. On the other hand, many instances of the L2PU need to be present in the simulation (up to an order of a few hundreds), which demands that the model of this component is not too heavy. The mapping of the *Region-of-Interest* data to the addresses of the data buffers also needed to be implemented to obtain credible data request patterns, reflecting the actual system. The utilisation of CPU caused by execution of the analysis algorithms had to be modelled as well. At the time when modelling was performed, the parameters describing the triggering algorithms were not yet available, and either rough estimates were used, or no processing was assumed - in the latter case the CPU utilisation in the L2PU nodes was observed, to determine how much CPU resources would be available for algorithm execution on each node, once the necessary data is delivered. A facility allowing to parameterise the execution of the algorithms (steps, CPU utilisation) was, however, also made available, yet never fully exploited.

In the multi-threaded model of L2PU (see Fig. 9.6) the Input Thread is responsible for receiving messages from the network; the messages are passed to the Message Dispatcher. The Message Dispatcher mechanism (Fig 9.3), used in the model, allows to de-randomize the message comming from data buffers, and deliver the requested data to the Worker Threads, that are executed in parallel. There is a dedicated handler for the LVL1 Results messages: it passes the LVL1 Result message to the LVL1 Result Queue. When the Worker Thread finishes processing an event and becomes free, it checks the status of the LVL1 Result Queue and fetches the event from the head of the queue.

9.3.3 DFM

The Data Flow Manager is responsible for traffic shaping and management in the ATLAS Data Flow System - a proper, detailed model of it needed to be provided for the studies performed with `at2sim`. Due to the fact that only one DFM node needs to be present in the system, and the importance of reproducing its functional behaviour as precisely as possible, I decided to implement a detailed model of the

Figure 9.6: Diagram of the L2PU model.



DFM, based on the fragments of code that were taken directly from the source codes of the real DFM application. This high level of detail in the model turned out to be needed so that the model of the system could pass validation tests.

9.3.4 SFI

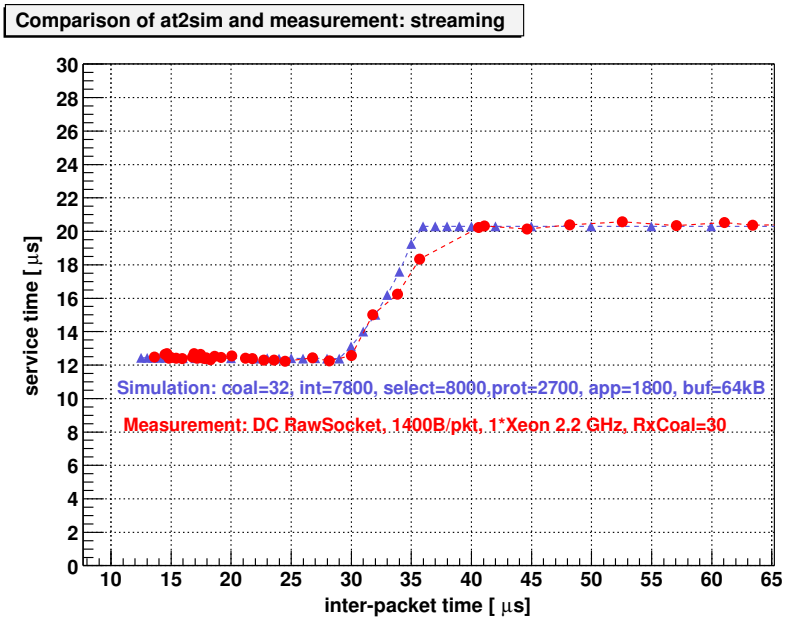
Similarly to the model of the L2PU, the model of the SFI needed to be sufficiently lightweight: tens of SFI nodes need to be present in the model of the final system. On the contrary, the functionality of the model was not as complex - there was no need for multi-threading or multi-stage processing in that case. The only important feature that had to be introduced (and tested) was credit-based data gathering. The output of the data to the SFO's were not implemented in the model - at that time this functionality in the real ATLAS Dataflow System was not yet ready either. Instead, as it was in the case of L2PU, the CPU Load on the SFI nodes was measured to estimate the amount of resources that are left for SFO communication.

9.3.5 ROB/ROS

Depending on the architecture of the ATLAS Dataflow System[90] being modelled, two types of the nodes representing read-out buffers are needed. For the network-based architecture[91], around 1600 simple read-out buffers (ROBs) need to be connected directly to the ROB-concentrating switches. In the bus-based architecture the read-out buffers are aggregated into complex, PC-based "ROS" (Read Out System) nodes executing multi-threaded applications responsible for data aggregation and retrieval from the ROBs. Initially the `at2sim` model was only developed for the network-based architecture. Later, the need for validation of bus-based approach appeared and the model of the ROS nodes was created. I was not involved in the design, development and parameterisation of this model, therefore I will not describe it here.

In scope of the studies of the network-based architecture an important role was played by emulators of the Read-Out Buffers, based on programmable FPGA-based Fast-Ethernet network tester, and a network tester based on programmable Alteon

Figure 9.7: Comparison between parameterised model predictions and measurement results: average CPU time needed to receive a single message in the network streaming test.



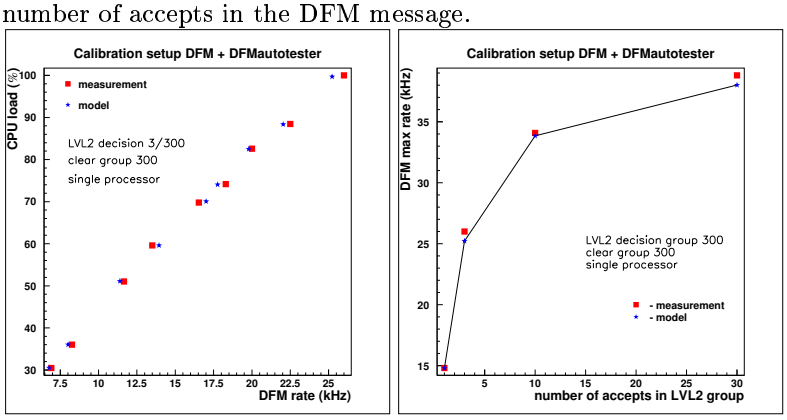
Gigabit Ethernet network cards. Both testers realised simple functionality of being able to send simple reply packets upon arrival of a data request packets. Important network scalability tests for the ATLAS Dataflow System were performed by means of the emulators[92], even before functioning Read-Out Buffers became available. It was therefore desireable to implement the models of both types of the ROB emulators in the `at2sim` to validate the model, i.e. to verify its predictions with the results of tests performed earlier by the group working on the network testing tools. Due to the large number of the ROB nodes that needed to be present in the system, the models of the nodes were simplified to absolute minimum. The use of the generic model of a node in `at2sim` was limited to the use of the network card model - the model of the operating system didn't need to be exploited.

9.4 Calibration and validation of the model

The parameterised models of system components needed to be calibrated using results from dedicated, small setups. The values of the parameters of the low-level communication models have been determined with simplified setups where maximum achievable message rates were measured, as described in chapter 7. The representative value that was used for the initial validation was the average time needed to receive a single packet in a streaming test. The value was calculated as a product of the CPU utilisation and the inverse of the rate of packets in the test, and was possible to calculate from the results of measurements and results produced by a model (for this purpose the very simple models of the nodes, based on generic models, were implemented). Once the callibration values are introduced, the model correctly predicts this quantity, taking interrupt coalescence and multiple stages of processing of the message into account (see Fig.9.7).

The callibration of the parameters for the models of the dataflow applications were performed as a joint effort of members of the modelling group, and developers of

Figure 9.8: Comparison between parameterised model predictions and measurements results for the DFM Application: CPU consumption vs DFM rate and DFM rate vs number of accepts in the DFM message.

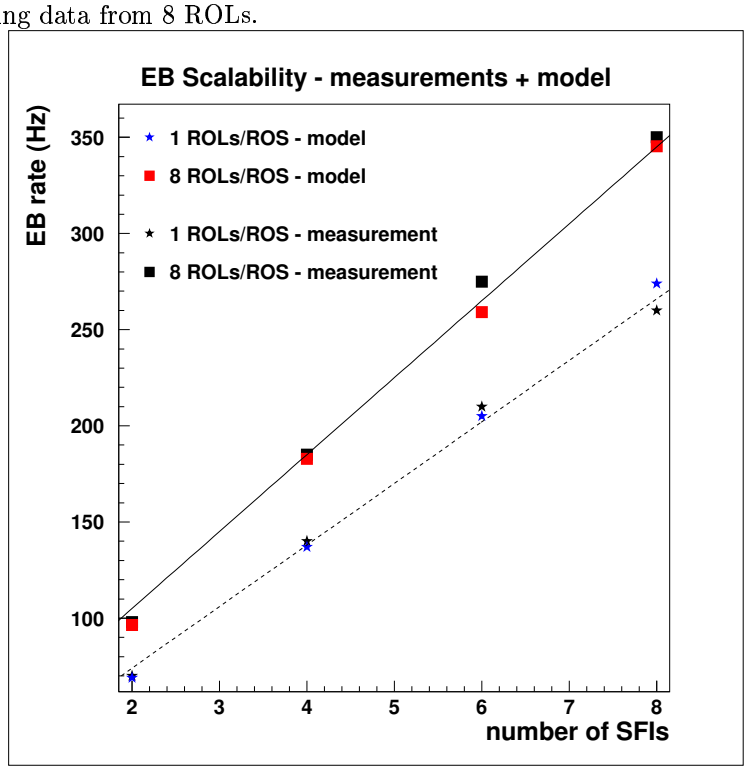


the ATLAS Dataflow System. Two approaches were used to obtain the callibration values: one based on instrumentation of the applications' codes with time-stamping statements, and one based on saturation methodology, where the application was interacting with special tester program, and the maximal rates were measured. The former method provided the values of direct measurements; large fluctuation in the values were however observed (caused by various sources, ranging from the precision of individual time-stamp, overhead of timestamping routines, and other activities in the operating system that could not be eliminated), hence statistical analysis of many results was needed. The latter method required that the values of the parameters were calculated from observed rates (and parameters set-up in the tests) - in this case statistical analyses didn't need to be performed: average values of the parameters over the time of the test were naturally obtained. Wherever possible, the two, complementary, methods were used to cross-check the results.

Similarly to the case of the callibration of the generic application (message passing), the callibration of the models of the nodes was followed by initial validation tests, that allowed to determine if the parameterisation (and level of model complexity) is sufficient. Typically, the tests consisting of "sweeps" of the parameters (of the real applications and the models) were performed, and comparisons plots were analysed. For example, the plot on the right side of Fig. 9.8 shows that the DFM model is sensitive to the number of LVL2 accepts inside the group of events received from the LVL2 Supervisor.

Once calibrated, the `at2sim` model has been validated by comparisons of its predictions with results of measurements performed for systems of growing complexity and the size of up to ten percent of the final system. One of the main purposes of these tests was to gain the confidence in the model being able to reproduce the scalability behaviour for the system. At that stage, the modeling was also used as an aiding tool for understanding the reasons for the behaviour of the system. For instance, the plot in Fig. 9.9 shows good agreement between model predictions and measurement results obtained with test setup aimed at testing scalability of the Event Builder part of the system. The maximum event building rate scales linearly with the number of SFIs. The intercept of the line fitted for buffers receiving data from a single Read-Out Link (ROL) (1600 network access points) is smaller than that of the line for buffers aggregating the data from 8 ROLs (200 access points), since a smaller number of request messages is required for the latter case. This results in a smaller fraction of the CPU capacity spent on generation of requests and increases the total number of events that can be processed per second.

Figure 9.9: The Event Building rate as a function of the number of SFIs for two readout scenarios: for buffers receiving data from a single ROL and for buffers aggregating data from 8 ROLs.



The results of many other validation tests, such as the ones presented in Fig. 9.10, ultimately convinced the group of modellers about the credibility of the `at2sim`.

9.5 Modelling results

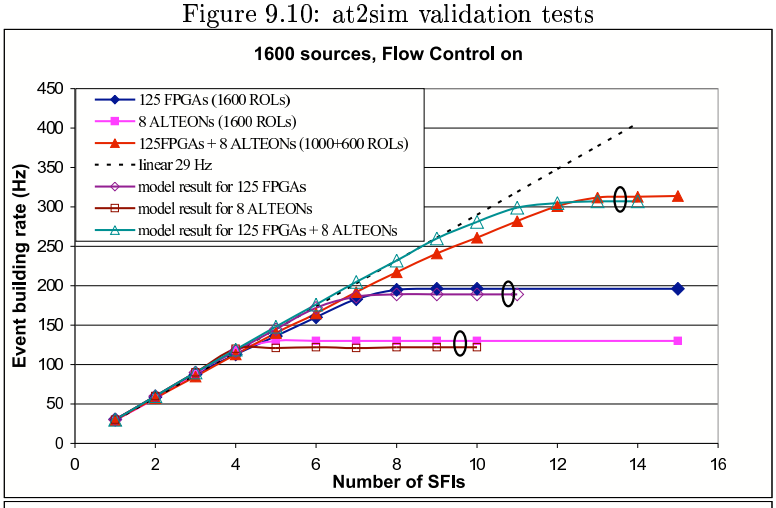
The validated `at2sim` model was used to produce predictions for various scenarios and architectures of the full-scale system. Recently, the credibility of the `at2sim` model was further improved in validation tests [93, 94]: `at2sim` was able to accurately simulate the 10% and 20% testbed setups. Then, it was used to produce further predictions for the full-scale system, confirming again the viability of the design: despite the fact that the L2 Processing Units did not execute real physical algorithms in the test, the system was capable to flawless execute its data-flow part, showing no bottlenecks in rates and throughput in the limits required by original design (up to 100 kHz LVL1 rate, 3% average accept fraction of LVL2 system).

The predictions based on modelling results (examples of which are presented in Figure 9.12) were an important milestone in the design and validation of the system - a milestone required by the Technical Design Report document [70].

9.5.1 Tests of various traffic shaping ideas

Models of the full ATLAS Dataflow System provide the possibility to study the effect of various ideas for traffic shaping on the development of queues in the system.

Figure 9.10: at2sim validation tests



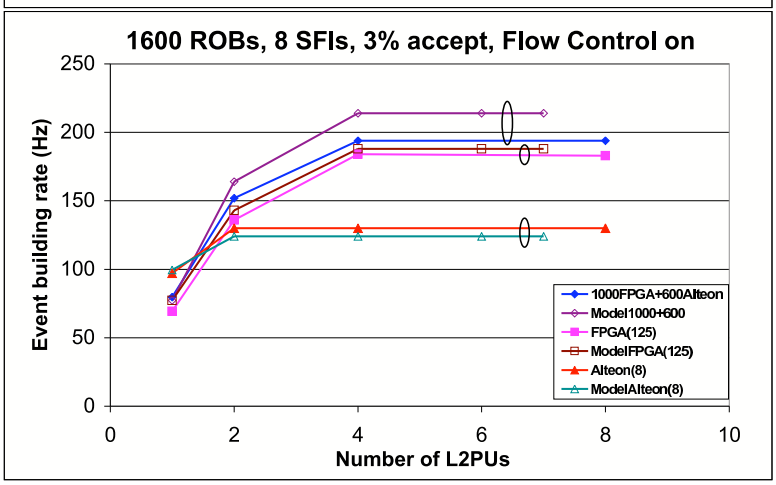
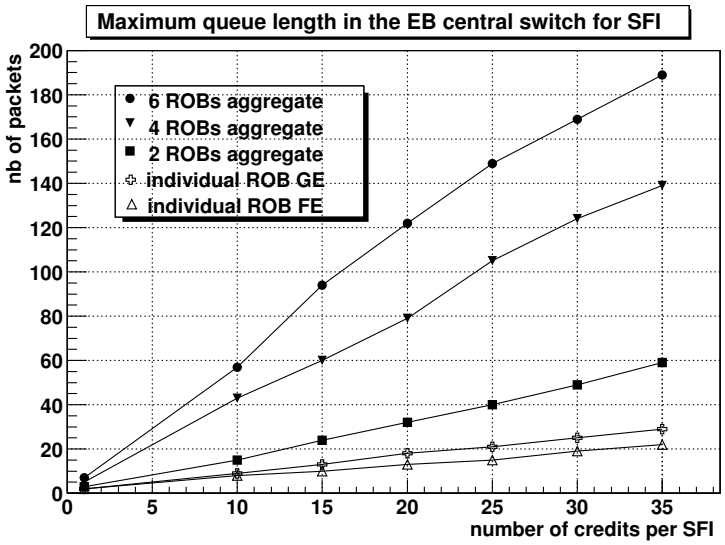


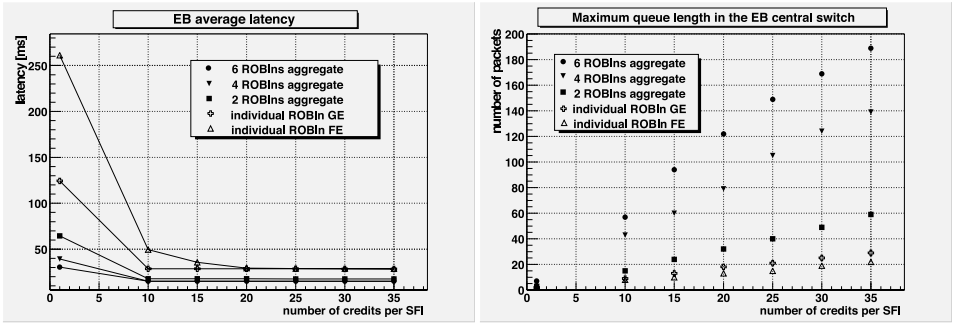
Figure 9.11: Central EB switch queue build-up for various scenarios for reading out the detector data



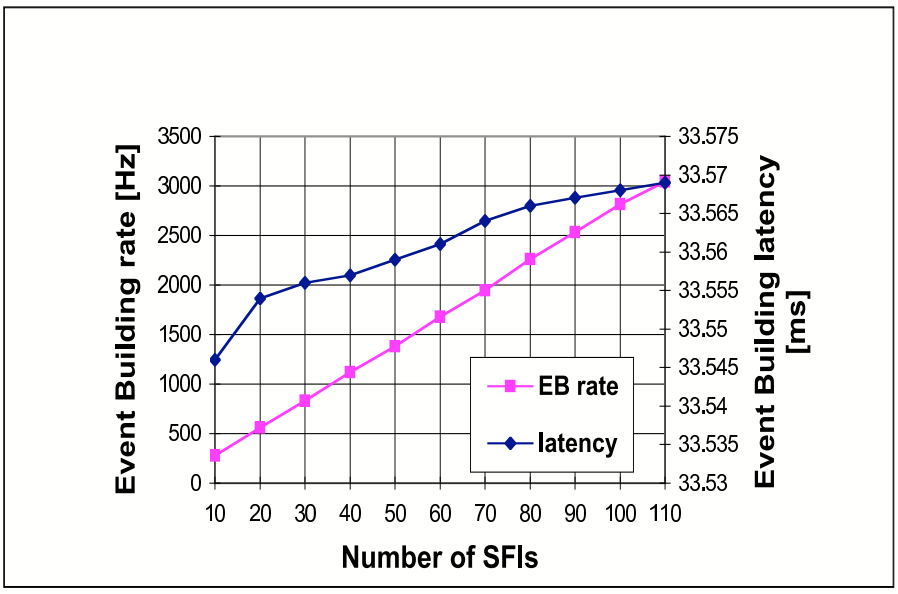
Limiting the sizes of these queues in the switches is essential to avoid packet loss in the network. This is important, as the data from lost packets have to be re-transmitted, which can result in a large amount of re-transfers, which, in turn may cause the whole system to crash. Information on the length of the queues in the end nodes makes it possible to estimate buffer sizes needed. The queue sizes can be limited by controlling the points in time at which packets generated by the end nodes are injected into the network, taking into account the buffering capabilities on the path to the destination.

In the event building subsystem, the traffic shaping can be achieved by limiting the number of outstanding requests for data sent to the detector buffers. Each SFI can generate only a limited number of requests for detector data (credits). The SFIs are allowed to generate new requests only after they have received replies on previously generated requests (i.e. when they regained a credit). By limiting the number of outstanding requests the SFIs can control the build-up of the queue with replies in the central switch. In Fig. 9.11 the maximum queue length in the central EB switch is shown as a function of the number of credits for different scenarios for reading out the detector data. Collecting data from buffers which aggregate detector data from more than one readout link (per readout link always one packet per event is sent) results in a proportional increase in the number of packets waiting in the switch for port availability. The event building rate is identical for all data points with more than five SFI credits - the rate is limited by the time needed to transfer fragment from all buffers (~2MB) over the Gigabit connection between the EB central switch and an SFI.

Figure 9.12: Examples of at2sim modeling results, as published in [70]



(a) The impact of the traffic shaping parameter (number of credits) on the performance of the Event Builder system



(b) Performance of the Event Building subsystem as a function of number of available SFI nodes.

Part III

Event Record

The complexity of the full-chain simulations favourize modular solutions, in which a set of functional blocks (routines or standalone programs) collaborate in building the event (the result of collision of particles) and simulating its development and the process of detection (decays of particles, their propagation through the detector). Already at the time of the LEP experiments an approach based on central data structure - event record - being used to exchange the data between the modules, and maintaining the state (and history) of the simulated event was established. A standard for the content and format of the event record, HEPEVT, was established in 1989 (it is discussed thoroughly in Subsection 11.1) and its implementation as FORTRAN77 common block was used as the standard interface between practically all modules in the simulation chain. The other de facto standard for event record, LUJETS, own event record of the Jetset and Pythia generators [40, 41] also became an alternative standard for event generators, due to popularity (and solidity!) of these generators. The limitations of the FORTRAN77 programming language, namely lack of compound data structures, required that the format of the HEPEVT event record is kept to assure compatibility and blocked the possibilities of introducing extensions, that began to be indispensable - problems due to data overolad in the too-rigid structure started to appear, and the HEPEVT lost its strongest virtue: being standard in its format and interpretation. The hope in the use of Object Oriented programming languages, unfortunately did not bring the solution to the problems: there is no commonly agreed C++ event record specification so far, and the existing ones seem to provide only partial solutions.

This part of the thesis is dedicated to the topic of the event record, and related data structures used in the full-chain simulations. It provides a solid reference of the most important event record standards, presents their strong and weak points. It also discusses the more fundamental problem of the viability of the central event-record approach, and tries to summarise the types of physical information that needs to be exchanged, and discusses the general data-structure issues. Ultimately, it documents my own approach to resolving the problem of a multitude of (incompatible) event records, with grammar violated due to data overload, namely the HepEventLib - a common "event record interpretation" layer which was used in the MC-TESTER tool.

Chapter 10

Event record and related data structures

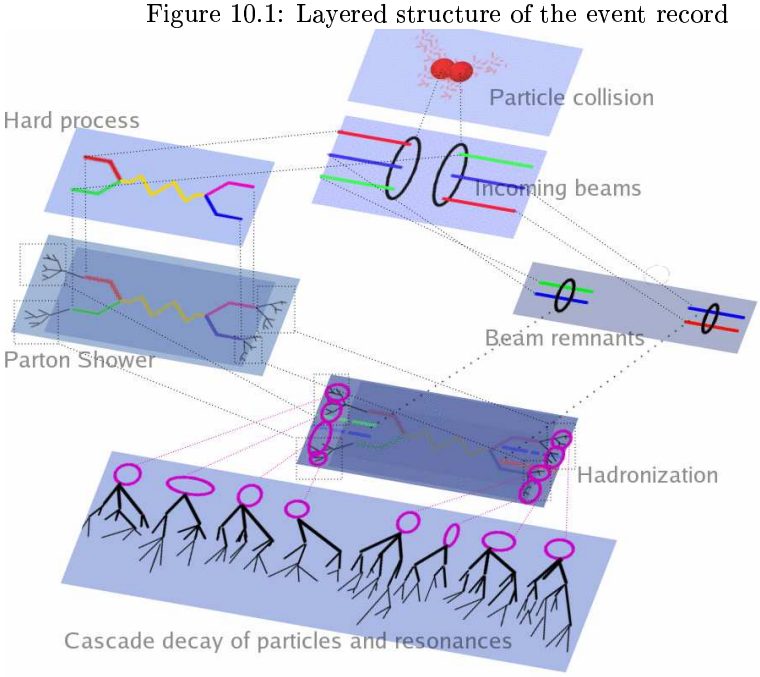
10.1 Introduction

Event record plays a central role in the event (and also full-chain) simulations for high energy physics. The flexibility of the solution from the time of LEP experiments, where each event was simulated and processed by a set of modules, which used the event record as their main interface and input/output routine was retained through the last two decades and, so far, it is the approach used by all experimental collaborations in HEP. The possibility of re-using existing simulation codes, validated over the ages, and comparisons across the boundaries of experimental collaborations only concerves the need for such approach and the need for a standard event record. In this chapter I would like to discuss the role of the event record and the requirements motivated by the needs of physics models. The constraints due to technical limitations of programming languages will also be presented.

Limiting the discussion of the event record to the technicalities of their implementation in programming languages seems to be one of the sources for the lack of established standard for C++ event record, and deficiencies of existing implementations. It is essential not only to master the technicalities of code development, object-oriented design and analysis, data structure and algorithm formulation, but also to really understand the requirements for such essential data structure, taking into account the needs and views of all modules that would need to use the event record. It is therefore prohibitive to limit the requirements to the ones specified by a few authors of general-purpose Monte Carlo event generator, and neglect the ones expressed by detector simulation, reconstruction or data analysis. Good understanding of requirements dictated by the physics of all these areas should be the entry point to any discussion. Then, once the form of encoding the physical information into computerized data structure is proposed, the interpretation for it needs to be established, and kept rigorously. Such process of "encoding" certainly puts constraints on the class of physics models that could be represented by postulated structure. The needs of the new models may require that the constraints are overruled - data structure or its interpretation may need to be changed, if they lack appropriate flexibility. Such situation may lead to catastrophic situations, such as the one of HEPEVT, where data overload practises based on ad-hoc conventions, defined (and imposed) by small groups of people are practised. All these factors would need to be taken into account before a new event record standard is proposed and developed.

In the following subsections we will discuss the requirements for the event record

Figure 10.1: Layered structure of the event record



Only event-generator related layers are presented; relations between instances of the same objects in various layers shown as dotted lines.

being the way of expressing physics information, and also being a computer data structure. We hope that this discussion may provide an initial set of requirements for specification of the future event record.

10.2 Overall structure of the event record

Event record is a container storing the information about the objects in the simulated or real, taken from detector, event. Typically it contains the information about particles, with their most important properties (fourmomenta, vertex coordinates), and the relations between them (such as the tree of decay cascade). The event record reflects also the history of the event - as such it is used as a "workspace" recording partial results of event generation performed by subsequent simulation modules, and provides series of sunsequent "snapshots" of the event record, with the same physical object being described at multiple stages of its evolution. Due to this multitude of uses, event record is typically a stratified structure involving objects, their properties and their various relations, which may be limited to the same layer, or link subsequent layers, see Fig. 10.1. The interpretation of the information contained in the event record, already at the end of the event generation step may be difficult: certain relations and objects may be artificial, and providing information that is unphysical (e.g. determining that a jet of particles originates from a certain quark) and reflects only the mechanisms used in the models. Nevertheless, access to such ("unphysical") information is vital for certain studies, in particular for optimisation and tests of reconstruction and analysis codes.

In the case of ATLAS offline software, there is no single event record data structure used; instead, various structures (such as HepMC event record conserving the snapshot of event generation stage, and native GEANT data structures) are kept together, with a convention to encode the relational information between them.

From mathematical point of view, event record is a complex graph. For the convenience of the algorithm interpretation it was however often forced into simpler structures such as a tree or a directed acyclic graph; these structures seem to constraint the physics requirements too much and does not seem to be a viable solution. In the case of the internal event record of PYTHIA generator, called LUJETS, the data structure is described as indexed table, with row number being used as object identifier; the tree structure is assumed, however, to encode relationships in decay cascade.

10.2.1 Requirements of stages of the full chain simulation

Event record specification should allow all modules (components) of the simulation to freely exchange the data with other modules, without the need for use of "shortcuts" or additional conventions for indexing the data. Due to sequential nature of the event generation, the structure of the event record is often optimised for the needs of modules of the event generators, in particular the most popular general-purpose generators such as PYTHIA or HERWIG, where the design is affected by the model of the event evolution (i.e. hard process, parton shower, model of hadronization performed within the same module). Other generators, that may be involved in the event construction (such as TAUOLA for particle decays, or PHOTOS for QED radiative corrections in decays) may usually locate the required information in the event record, and enrich the event record with the results of their simulation - the type of information exchanged seems to be similar: the relationships in the decay cascades and the four-vectors of particles. The simulation is more complicated for instance for the case when the generation of the hard process and the parton showering are done separately: the complete information that needs to be passed cannot be expressed by means of simple relationships and momenta fourvectors. The step that follows event generation in the full chain simulation is detector simulation. Between the two, the approach and way of expressing data need to be changed: one refers to hits and energy deposits rather than particles; at the reconstruction steps one associates the hits with tracks - another class of objects. Ultimately, the analysis tries to identify particle properties using tracks, and refer them to the original particles produced at the event generation step. A viable event record standard should make it possible to store and retrieve all kinds of information from all these steps, and provide ways to store the information about the associations (links between the objects) across the steps of the simulation (i.e. storing MC Truth information).

10.2.2 Physical properties of objects (event generation stage)

Let us now concentrate only on the properties of the objects taking part in the event generation stage; the list would need to be completed with the missing properties of all objects taking part in detector simulation, and possibly analysis step. The complexity of these subsequent steps would only complicate the picture presented in this chapter; on the other hand, help from detector simulation experts would definitely be needed to prepare a complete list.

Momentum and mass

The most important property of a particle is its momentum four-vector, the components of which enter directly the formulae in theoretical models (matrix-element, phase-space), and they are the observables measured in the experiment as well. The energy-momentum is usually treated as an object of its own category, because it either takes part (as a whole) in calculation of other quantities or is modified as

a whole by common operations, such as Lorentz transformations. Therefore typically it is hadled and stored in form of a vector with four components $(E, p_x, p_y p_z)$, optionally completed by the fifth component being the mass of the particle.

Despite being the feature determining the type of each particle, the mass is an ambiguous quantity in simulations, in particular for the cases of wide resonances and off-shell particles. Multiple definitions, interpretations and values for the mass may be used, depending on the context. At first, it may be equal to the square of the momentum four-vector, i.e. $M_{\mathrm{on-shell}} = \sqrt{E^2 - (\overrightarrow{p})^2}$. One should note that this assumption (of the particle being on mass shell) is often used in four-vector operations such as boosts. For the off-shell virtual particles (internal propagators in Feynman diagrams), which may be present in any event record as well, the mass does not need to be equal to on-shell mass (the propagator typically has singularities on the mass shell) and the difference describes the virtuality of the particle (the amplitude for the process diminishes as the mass goes more and more off-shell).

The other value for the mass may come from the PDG tables and expresses the value of mass measured in exepriments. This information is also accompanied by the width Γ or lifetime τ of the particle and accompanying measurement uncertainty.

Vertex

The information about the coordinates of production and decay vertexes for the particles is another part of the information in the event record. This information is extensively used in the detector simulations - typically vertex coordinates of the final state particles at event generation stage form the input data for detector simulation stage. Please note that the vertex information is not sensible in all contexts: in particular for very short living particles providing the coordinates of production/decay vertexes does not make sense (they are practically equal to the coordinates of the interaction point); for simulations of hard process, parton shower or hadronization they have no physical sense.

Historically, in HEPEVT event record, the coordinates of production vertex were particle properties stored together with other information concerning the particle. Alternative approach is used for instance in HepMC: vertexes and particles are stored separately: each particle has links to its production and decay vertexes, and a vertex contains the list of incoming and outgoing particles. In such approach, additional information about each vertex may usually stored (e.g. spin density matrices).

Particle status code

This is an auxiliary portion of information used internally by event record data structures, typically used for management of event history information. Usually it has a form of a "status code" associated to each object (particle, documentation entry in the event history), which tells whether the particle is stable, decays, it is a parton, etc. The encoding of this information is generator-specific, yet some standards are defined i.e. for HEPEVT. This type of "general-purpose" data tag could probably be avoided by proper design of data structure: the conventions for status codes might be replaced by appropriate, well-defined structures dedicated to storing relevant data and relationships.

Spin information

The models involved in event generators typically involve calculations that eliminate the spin degrees of freedom, by performing appropriate summing and averaging in the differential formulae. In majority of analysed physics processes such approximation was sufficient and greatly simplify the calculations. It was only the few

cases of spin-sensitive experiments that use polarised beams, such as production and decay of the τ-leptons, where the effects due to spin correlations may affect the observables, and needed to be taken into account in simulations and the data analyses. For the phenomenology of the future colliders, however, proper treatment of spin correlations will be vital for studies of supersymmetric particles; it may also play role in determining the CP quantum numbers of the Higgs boson [2, 95, 96].

The importance of proper spin treatment in event simulations are discussed in length in [97]. The currently applied approach to modular building of the event, due to assumption of particle production and decay mechanisms being separable is in contradiction with the most basic principles of Quantum Mechanics, depicted for instance by the Einstein-Podolsky-Rosen paradox: [1]. The phenomenon of quantum state entanglement, which is the origin of the problem (and the paradox), is however not only a theoretical prediction: quantum cryptography has already found commercial applications with first "turnkey" solutions being already available [98, 99, 100]! To assure proper treatment of spin, the simulation of production and decay processes needs to be performed at the same time in a "monolithic" generator (or some intermediate, approximated solutions need to be used).

A few approaches to express the spin information exist, the most general being the use of spin-density matrix, $\rho_{\nu\mu}$. An often neglected, yet vital issue is the definition of spin quantisation frames, and the appropriate methods to perform Lorentz transformations on particles, taking into account spin information. A general solution to this problem has been proposed in [101], and the method used successfully in constructing a high-precision Monte Carlo event generator demonstrated in e.g. [102].

Another scheme for the treatment of the spin correlations, in the QCD parton shower Monte Carlo generators, was proposed in [103]. This mechanism was extended in [104] to deal also with decays of heavy particles and it is used in HERWIG[33]. The solution that was used is however based on the narrow-width approximation, where one can decompose the propagator into a sum over physical polarisation states. It is wrong to assume the general correctness of this assumption: for certain decay channels of the τ leptons, the width of intermediate resonances are comparable to their mass. On the contrary, the TAUOLA generator is capable of generating the production and decays of τ's in appropriate way.

An intermediate, approximated solutions for restitution of spin information in the events simulated using generators that do not treat spin information is sometimes also possible. Even though they provided only an approximated treatment, they already provide more precise modeling of physics phenomena. An example of such approximation is the TAUOLA Universal Interface, described in [2] and also briefly in chapter 14.

Colour flow

Hard-process, parton-shower and hadronization generators need to exchange the information about the flow of the colour. The lack of the possibility of storing the colour-flow information was the main reason of overload of the HEPEVT event record in the HERWIG generator. A scheme for encoding the colour flow was proposed e.g. in [105] as a part of "Les Houches Accord".

Particle properties tables

A well-defined part of every simulation is the initialisation of physical constants; for event generators this would imply the values for particle properties such as masses, widths, but also decay channels with their widths or branching ratios, etc. Certain particle properties, such as electric charge, or spin, may be expressed precisely

using discrete values; others, such as mass, life-time and branching-ratios need to be measured in experiments - their values are specified with certain precision and measurement uncertainties, which evolve (precision improve) as more and more precise results are available. The list of the values of properties for all know particles is managed by the PDG group; it will be presented briefly in section 11.2.

The choice of the values for the parameters is however not a straightforward task; hardly ever taking the tabularized valued from the PDG list is appropriate... First of all, for precision measurements, experimental collaborations often maintain their own parameterisations or form-factors (i.e. for hadronic currents), and initialisations of their parameters expressed as masses, decay widths, that are based on the fits to the data to the physical model. Blind choice of the values for the parameters, for instance taking the average values from the PDG list[1], may lead to unpredictable effects and results of no physical meaning! Another example is the use of generator to simulate very rare decay channels, which were measured with very small precision, or have not been observed at all - sensible information about the masses and widths are not available at this case.

Other information

There are other types of information that is used at the event generation stage, in particular the information related to the treatment of multiple-collision events, and binding the showering generators with matrix-element generators. Among others, the values of the coupling constants needs to be available for the generators which collaborate in the generation of the event.

The other class of information is internal state of the physical model implemented in the event generator; for instance, the event record implemented in ThePEG framework stores the links to objects representing matrix elements or process tables.

Other information related to detector simulations

The detector-simulation packages, usually based on the GEANT framework, treat the event record as an input for their processing. Firstly, the information about undecayed particles and the vertexes needs to be transferred to some internal representation: in the case of GEANT these are so called "KINE" stack and Vertices data structures. A set of routines is usually made available for that purpose. We will not discuss the details of the translation and the data structures here. We note, however, that the information about the relations of the particles simulated in the detector to the particles from the event record needs to be preserved, in particular for the algorithm studies, where it is available as the so called "Monte Carlo Truth".

10.2.3 Relations and their encoding in data structures

One of the critical issues for the event record is encoding of relationships between the objects. The problems that need to be resolved are not limited to the technical level, but also involve understanding of the interpretation and constraints put on the class of physics models that may be represented using the data model being proposed.

From the technical point of view, encoding relationship between objects requires that some referencing mechanism is used. In the case of FORTRAN event records, the typical solution is based on a common block with a set of arrays, the index of which is used to refer to the properties of the objects. The same indexing number

[1] The choice of one value out of the list of available ones is not trivial and requires profound knowledge of underlying model that was used! Even the mass of Z differs by 27 MeV depending on the choice between fixed and running width!

may play the role of a pointer, or a reference, to implement "links" of relationships between the objects. The methodology of designing the data structure that involve implementing relations, where the data storage is limited to arrays (or tables) is in fact very well established, also on the mathematical basis, as it is the key element of theory and practise of relational databases. Despite the fact that structure- or object-like extensions become available for the relational databases (nested tables, hierarchical queries), the most basic key-based relation approach and data model is the preferred way of design and use of the databases.

For the languages such as C/C++, which natively support pointers, references and complex data structures, the design of the data structure is not so constrained and allows the use of abstractions to represent the data. Introducing relational links between objects (or its parts), aggregations and containment relations becomes natural using these mechanisms. The problems arise, however, in the software architectures that rely on objects persistency[2] (as it is, for instance, for the case for the ATLAS offline software) - access to referenced data (following relational link), or any modification of the relation may then appear to be very heavyweight and resource-consuming operation, because it is not limited to simple de-reference of the pointer variable, but usually requires back-stage activities, such as finding the proper object in the external storage space, loading it to the memory and updating the addresses stored in the pointers. Design and implementation of a robust structure requires balancing the need for abstraction and the performance.

Let me now turn to the discussion of the relations in context of requirement of physics models. The set of requirements presented below is, again, not supposed to be complete - in particular, the relations that need to be established at the level of detector simulation, reconstruction and data analysis would need to be identified with the experts. The relations that I present are mainly the ones required by event generation step.

As outlined already in the previous subsection, the relations that need to be encoded by event generators contain the information that either has a direct (or indirect) physical meaning, or reflects the specifics of the model implemented in the generator. The notion of layers, representing the history of the event evolution is omnipresent in the event record and requires the links that identify instances of the same object in various layers, at various stages of the evolution. Encoding subsequent steps of parton shower evolution is amongst the most obvious use cases here: splitting partons

Similar links, introducing relations across the layers, are needed to bind the representations (or: views) of the same object in various stages of the simulation; the most prominent example being the relation between particle in the generated event, and hit in detector simulation, or series of hits in detector and reconstructed track.

The number of layers in the event history, and hence the number and nature of the links that reflect the relations cannot be pre-determined. In particular, it depends on the physical models used and their implementation by particular event generators. Some of them, such as PYTHIA, may implement multiple ways of storing the data in the event record, depending, for instance, on the detail level of event history information requested.

The relation that originally was the only one implemented in the HEPEVT event

[2]Object persistency mechanisms allow to store the objects (data structures) in the persistent storage (disk file, database), then re-store the data on-demand. It is usually implemented as a part of low-level infrastructure, and usually heavily affects the architecture of the project (in a sense: there is no need to implement the dedicated input/output routines for all data: a "snapshot" of data may be made permanent, then re-loaded any time later). Typically, the objects that are persistent are indexed/classified in some way, or organised in a database, which allows to bring up to the memory only the requested ones (or the specified versions of them).

record was mother-daughter relation in the decay (and production) of particles. In the cascade decay process, the particles naturally organise in a tree structure: a non-stable particle decays to two or more particles. The assumption of a tree structure would lead to an effective way of storing the relationship information. By use of topological sort (see next section), one might arrange the nodes of a tree in a list in such a way that all "child" nodes of a particle are stored contiguously in the list. It would however be sufficient to store the reference to the first and to the last child in the list; obtaining the list of all children would then require getting them linearly from the list, starting from the first child and finishing on the last. That would also make the search/iteration algorithms very easy to implement (a simple DO-loop in FORTRAN).

The tree structure of the decay cascade may enforce in natural way the energy-momentum conservation principle in decay vertex, and, as such, it seems to be an "elegant" solution from physics point of view. The same tree structure, however, becomes insufficient to express... Feynman diagrams, if more than one diagram contributes to the process!

Despite its elegance, robustness and ease of use, the simple tree layer representing decay cascade is, however, far from being sufficient to encode the information from even a simple, complete event that (typically) starts with (two) particle collision. That is why the structure of the mother-daughter relationships was extended already in the HEPEVT event record: apart from information about children, each particle may contain the information about up to two mother particles. Storing additional pointers to parents not only allow to encode $2 \to n$ process topologies, but also made the navigation in the upstream direction easier: the pointer to parent particles were readily available, and the linear search through the list of all particles to find the parent of a particle could be eliminated. Unfortunately, at the same time, the source for potential data inconsistency was introduced. The mother-daughter relational information was stored in redundant way: for a process $A \to B, C$, not only A stored the "daughter" relation to B, but also B stored the "mother" relation to A. The initial tree structure, being a directed, acyclic graph was converted to a more general graph, with no possibility of enforcing relational data consistency: the way was open to create loops (cycles); the advantageous, simple and performant algorithms for tree searching and iteration could not be guaranteed to work anymore. Deliberate overloading of relation-encoding data redundancy, to fit additional information in the too rigid, inflexible HEPEVT event record led to incompatibility problems that we suffer from to these days (i.e. the use of PHOTOS or TAUOLA with events produced by HERWIG seems to be almost impossible, despite they all communicate using HEPEVT standard, which is abused in HERWIG).

An alternative, and less constrained, approach to encoding of the decay cascade relations is to separate the particles and vertexes separately, which allows to express vertex for any $n \to m$ process. It requires that each particle requires relational links to its production and decay vertexes, and each vertex containing links to the incoming and outgoing particles. Introducing the additional level of abstraction, namely the vertexes, makes the data model better reflect the physics model; on the other hand it complicates the algorithm that need to traverse the structure (for instance, getting the list of particle children requires involvement of the vertex object at the intermediate step).

The encoding of colour information requires another type of relation: flow. The flow relation need to link arbitrary number of particles, two of them being "dangling ends".

The method for encoding of spin information, in particular in the context of spin correlation treatment, depends on the approximation and approach used in the model of physical process. As mentioned in previous subsection, full treatment of the spin correclations requires that the process of production and decay of a particle

are simulated together. In certain solutions, however, such as the one proposed in [104], spin density matrices need to be associated with vertexes.

10.3 Event record as a graph

The field of mathematics on which the structure of the event record (and data structures in general) is to be discussed is graph theory. In this section let me summarize the most important terms and facts, in particular the ones that refer to trees, which are a sub-class of graphs, on which the design of event records relied in the past, and still rely at present. The summary is based on information and definitions from [10].

Informally, a graph is a set of objects called vertexes, connected by links called edges. Directed graphs have their vertexes connected by directed edges, called also arcs or arrows. A variation of the directed graph is the oriented graph, which can only have one edge joining two points. A directed graph may have loops, that is, edges where the start and end vertexes are the same. By definition, loops are forbidden in oriented graphs. If each vertex is given a label, then the graph is said to be a vertex-labelled graph.

A tree is a connected, undirected graph with no cycles - any two vertexes of a tree are connected by exactly one path. A disjoint union of trees is called a forest. In a rooted tree, one of the vertexes is designated to be a root. Rooted trees are one of the most important data structure in computer science.

A directed acyclic graph, also called a dag, is a directed graph with no directed cycles. A dag can be considered as a generalisation of trees, in which certain subtrees can be shared by different parts of the tree. A dag may be expanded in a simple way to a forest of rooted trees.

One of the most important properties of a dag (and a tree) is the possibility of performing topological sort: an ordering of the vertexes such that each vertex comes before all vertexes it has edges to. This property is heavily exploited in the construction of FORTRAN event record data structures, which use indexed arrays to store the nodes, and encode mother-daughter relationship e using ranges of indices. Many algorithms, in particular the ones performing search in data structures become simpler when used on dags instead of general graphs.

The process of topological sorting, according to [35], may be performed on the items over which a *partial ordering* is defined i.e., where an ordering is given over *some* pairs of items, but not among all of them. In general, a partial ordering of a set S is a relation between the elements of S. It is denoted by the symbol $\prec$ ("precedes"), and satisfies the following axioms for any distinct elements x,y and z of S:

$$(x \prec y) \wedge (y \prec z) \Rightarrow x \prec z \quad \text{(transitivity)} \tag{10.1}$$

$$(x \prec y) \Rightarrow \sim (y \prec x) \quad \text{(assymetry)} \tag{10.2}$$

$$\sim x \prec x \quad \text{(irreflexivity)} \tag{10.3}$$

The problem of topological sorting is to embed the partial order in a linear order. Properties 10.1 and 10.2 of partial ordering ensure that the graph contain no loops, which is the prerequisite condition under which such an embeding in a linear order is possible.

The trees as data structures and the algorithms to perform operations on them are presented in book [35], the title of which: "algorithms+data structures = programs" may be considered as a motto that joins parts *III* and *IV* of this thesis. In the book, edited already in 1975, the author: Niklaus Wirth, who is considered

as one of the pioneers of the computer science and the art of programming, points out:

> "(...)We shall not further discuss other possibilities of representing trees in systems that lack a dynamic allocation facility, for we assume that programming systems and languages including this feature are or will become commonly available" [[35], pages 194-195].

The conjecture of popularity of languages with dynamic allocation expressend by prof. Wirth already in 1975 indeed came true. On the other hand, we face the enormous and priceless heritage of FORTRAN-based scientific software, and the limitations in expression of data structure imposed by language constraints. 15 years after this sentence was expressed, HEPEVT specification, with array-based implementation of trees was standardised; 30 years after it, experimental collaborations that prepare to the LHC startup still rely heavily on the codes that use HEPEVT as their main interface...

The problematics of advanced data structures covered in the book considers mainly the binary trees, i.e. the trees in which a node may have at most two branches (such structures are widely used e.g. in sorting algorithms; the other variant presented there, B-Trees, are used commonly in databases). According to the nomenclature of prof. Wirth, the data structures that we would consider to be best suitable for event-record would fall into the category of "multiway trees", in which multiple relationships between elements can be defined. Let us quote another excerpt from the book:

> "(...)Such a structure quickly grows into a complex "relational data bank," and it may be possible to map several trees into it. The algorithms operating on such structures are intimately tied to their data definitions, and it does not make sense to specify any general rules or widely applicable techniques." [idem]

Again, this quotation could be seen as a prophecy for the problematics of the HEPEVT event record; its evolution, motivated by physics modeling needs, required that additional relationships were encoded in the event record, violating the original properties of the structure, i.e. partial ordering, and rendering the universal algorithms for search/modify operations (i.e. the operations based on topological sorting) impracticable!

10.4 Event record and common operations

Let us reprise (and rephrase) the motto from the previous subsection: to constructs programs, one needs data structures and algorithms that work on these structures. Having already discussed event record as a complex data structure, with a multitude of encoded relationships, I will proceed with discussion of the most common operations (or: algorithms) to be performed on the event record.

General operations related to data structures may be classified to two categories: data manipulation (filling the structure with data, modifying existing data) and data retrieval (locating requested information in the structure, "find-type" operation, navigation through relationship-encoding links). Similar classification may be observed in database systems, where typical operations involve data manipulation (SQL-DML Data Manipulation Language statements such as INSERT, UPDATE, DELETE) and data queries (SELECT statement of SQL).

Conceptually, data manipulation operations seems to be less complex. In a flexible, dynamic structure adding the data (enriching the information) is typically an

elementary operation: the space for the data may be automatically allocated in the data structure; establishing relational links usually requires to involve some data search operation to establish the "targets" (objects with respect to which the relation is encoded). Removing a part of data also seems to be elementary operation; care needs to be taken, however, to properly invalidate all the relations that lead to deleted information. Modification of the data typically involves data search operation, followed by elementary "update" operations; this may require re-allocation of storage space and update of relational links.

Data manipulation operations become much more complicated in the systems without dynamic data structures and pointer/reference mechanisms. I will use HEPEVT event record as an example, and consider a typical operation that needs to be performed by PHOTOS Monte Carlo, i.e. adding a photon as a child in a decay of a particle located in arbitrary place of decay cascade. Due to the requirement for topological sort of daughter particles, the photon cannot be placed in any free place of the event record (e.g. appended at the very end), but needs to be put at the place that is adjacent to the where the children of the decaying particles are stored. This requires that firstly appropriate space in the tables is made , to fit the new photon; in HEPEVT, which uses arrays to store objects and array indices to encode relations, this requires quite complicated, multi-step operation. At first, all the particles with indices larger that the one of the decaying particle need to be "moved" one place further: this requires that the data for each particle that is stored at position $n > m$, where m is the position of decaying particle, is copied from to position $(n+1)$; then all the indices in the event record, that pointed to the particles that were moved need to be updated; ultimately, the data describing the photon may be put in the prepared place, and daughter pointers of the decaying particle may be updated. Certainly, for a significant number of simulation modules taking part in the evolution of the event, the objects are appended at the end of the event record, rather than inserted in the middle, hence the renumbering operation is not needed and the operation is much simpler.

Data manipulation operations on the event record are not, however, the most crucial ones - the main role of the event record is to be capable to provide information to other modules, in a consistent and performant way. Data search and retrieval operations, and traversing the relationship links need to be performed much more often, in particular at the downstram steps of the simulation chains (reconstruction, analysis), when the contents of the event record becomes reach and the topology of relationships - complicated.

To realise data search and retrieval operations that need to navigate through sets of data items connected by relations it is convenient to adopt the concept (and abstraction) of iterator[3]: a mechanism that allows to sequence through the elements in a set, according to some defined iterating rules. In the context of databases, iterators are often refereed to as cursors, whereas in programming languages they are thought of as a generalisation of a pointer. Two types of operations may be performed on any iterator: dereferencing (or element access), which gives the access to the data being pointed to by the iterator, and one or multiple traversal operations, which modify the state of the iterator in such a way that it points to other element. Typical traversal operations are, for instance, the ones to traverse to the previous or next element in the ordered list. In procedural languages, such as FORTRAN, iterating operations are performed by means of indices, which play the role of the iterators, and "for" or "do" loops, which implement the traversal operation. In the object oriented languages iterator mechanisms (which are often discussed together with container objects, such as vectors, lists, etc) are a part of the language specification (Java) or standard library (STL for C++). The operations

[3]see also the definition in [10].

related to navigation through a variety of links in the event record may therefore be thought of as iteration on the event record; for instance, an implementation of an iterator may be created, that will be used to traverse the list of children of a decaying particle, returning them one by one. Other iterator may, for instance, traverse only the non-decaying (stable) decay products of a given particles (i.e. at the backstage, the traversal algorithm implemented in the iterator would traverse the decay tree down to the level of leaves). The iterator mechanism could also be implement to traverse the event record in "vertical" direction, to navigate across its layers, to return all the instances of the same particle at all points of event history. The abstract of iterator may be also used to hide the actual details of internal organisation of the data structure; for instance, the iterators written for HEPEVT in scope of HEPEventLib project (see subsection 11.6.9) are capable of resolving overloaded data and conventions used by generators such as PYTHIA or HERWIG, and allow to traverse decay cascade in correct and easy way. High abstraction-level Iterators, with profound physical content, could also be thought of, for instance to eliminate double-counting in matrix-element-to-parton-shower interfaces.

Complementary to iterators are the mechanisms that perform searches for data of particular characteristics. In the context of event record, a variety of such algorithms, typically of physical meaning, need to be implemented. For instance, in PHOTOS, one of initial steps consists of identifying all decaying, charged particles in the decay cascade, which are later treated as potential points of photon emission; in TAUOLA event record is searched to identify undecayed τ leptons pairs. In the studies of performance of reconstruction and physics analysis codes complex algorithms that need to traverse many layers of the tree often needs to be created. Particularly difficult to implement (due to unsettled gramar of existing event records) are for instance the one that gives the answer to questions such as "what is the flavour of a quark (in the hard process) that results in specified jet that is responsible for specified energy deposit". The algorithm capable of providing such, in fact "non-physical" answer is possible to implement, yet it needs to be aware of the way to traverse various types of relational links, including dereferencing the boundary between detector simulation and event generation, navigating upstream the decay cascade, resolving the relations in hadronization and parton shower, then cross the modeling boundary between non-perturbative parton shower and matrix-element based hard process. Due to enormously rich physics content (and programming effort), implementation of such algorithms may usually be a subject of a separate, independent project.

The availability of appropriate architecture for iterators and search algorithms (and the iterators and algorithms as well!) would certainly make the implementation of physics algorithm much easier. The amount of code that implements such algorithms in generators such as TAUOLA or PHOTOS could be greatly reduced, which would make the implementation of their physics models significantly clearer.

The index-based data search/modification operations in PHOTOS are particularly vulnerable (ans sensitive) to inconsistencies in the gramamar of the event record data. They are the main source of problems with interoperability of PHOTOS with other generators, and endless efforts with debugging of problems of this nature in various environments. For that purposes the code of PHOTOS is equipped with large portion of tedious code responsible for verification, safeguarding (and fixing) of the gramatics of HEPEVT, so that subsequent loop-based event-record operations may be safely performed. The use of a *reliable* and flexible "intermediate layer" of event-record related algorithms and iterators, in place of PHOTOS implementation, would certainly make the code more reliable, easy to maintain and inter-operable (e.g. the problem with non-compatible gramatics of the event record could have been trivially eliminated (and hidden) completely, at the level of iterators implementation). Such translation for PHOTOS has already been envis-

aged[4] (and performed [56], yet without the benefits of iterators) and it is believed to be able to resolve the majority of inter-operability issues. The experiments with iterator-based approach, performed in scope of HEPEventLib prototype (see below, Subsection 11.6.9) demonstrated its flexibility - the use of HEPEventLib as event-record access layer in MC-TESTER allowed to assure inter-operability with a variety of flavours of event records (including HERWIG's format of HEPEVT). Implementation of a layer of algorithms (methods), closely related to the data structure is natural in Object Oriented approach: event record should not be limited to a data structure anymore, but should contain related functions as its part. In the existing proposals and implementations of C++ event records (HepMC, ThePEG) the concepts of iterators and algorithms are certainly employed, yet their use seems to be limited to a quite low abstraction, "technical", level, which exposes (sometimes overwhelming) richness of C++ language constructs and concepts, yet does not provide enough abstraction to fit the language of physics.

10.5 Remarks and open questions

In this subsection let me pose and address a few open questions and remarks concerning event records, that I find particularly important or interesting.

Data exchange or internal state representation

The use of event record is usually considered in two context: as an interface allowing to exchange the data between modules of a complex simulation (which is its initial role), or as a representation of the state of the event generator. The issues related to the former were the main topic of this chapter; the latter, however, seems to be a common practise of the authors of event generators, who introduce custom conventions and extensions of the data model, so that the state of the generator at various steps of event generation may be documented in the event history. The event history indeed is supposed to contain such information, which certainly needs to be model-specific, yet it should not violate the rules of the event record! An example of flexible organisation is the PYTHIA generator, which has its own "event record" data structure (LUJETS/PYJETS) to store the key information about the event being generated, and a set of routines to translate the data from/to the standard event record (HEPEVT).

In general, the use of the event record as ultimate storage of generator state only leads to problems: too simplistic, generic data model leads to data being "forced" to the data structure and data overloading; often model-specified, detailed information is stored while the structure is violated - the state of the generator is stored, but the information is practically unusable (due to rules violation) by other modules of large simulation (i.e. at detector simulation and data analysis).

Use of dedicated "layers" in the event record

Certain generators, such as PHOTOS, need to modify large parts of the event record: whenever a photon is added the kinematics of all particles in a "branch" of decay cascade is modified; PHOTOS performs the operation on "living organism", i.e. it does not preserve the "historical" information about the state of the event before its run, but "overwrites" (or simply: modifies) the event record. For event records such as HEPEVT, which historically were limited in the size, such approach was probably the most pragmatic. In the new applications, where the amount of

[4] The only blocking factor remains the lack of implementation of commonly agreed C++ event record standard.

available memory and processing power, and availability of dynamic data structure is not an issue anymore, the use of separated "layer", documenting the activity of PHOTOS would probably be more appropriate.

Use of conventions

One of the possibilities of encoding additional information in data structure of limited flexibility is to use conventions. For HEPEVT, conventions were used to document event history, or to force the colour flow information into mother-daughter relationship pointers. In the case of HepMC used in ATLAS, conventions are used again to store the information about layers of the event history. Such conventions are typically defined as "ad hoc" solutions, and do not cross the boundaries of experimental collaborations; such short-term, local solutions may cause significant problems when analyses of integrated data from multiple experiments are performed, or the results need to be compared with the results of other collaborations.

Validity and viability of event-record approach

In the light of component approach to scientific simulations, as discussed in Chapter 4, the use of the approach with central event-record in the future[5] HEP simulations should be questioned. From technical point of view, the component-based approach requires that any communication (data exchange) between modules is performed by means of well-defined interfaces (the definition of which needs to follow the definition of the architecture). Potentially, multiple standard for architectures may exist in parallel (i.e. each experimental collaboration may maintain its own architecture) , and the components might be freely exchanged and reused at the price of implementing data-exchange interfaces to the architectures that want to use them. At least a single, universal architecture, with clear specification of the interface, should however be firstly defined, to provide a baseline, standard implementation; its role would be to replace the "standard event record". The question of design of the architecture would need to be addressed very carefully, so that enough flexibility and functionality is provided for physics algorithms.

From the physical point of view the approach with central event record, and simulation modules subsequently generating the event is also questionable, when spin degrees of freedom need to be treated properly in the model; as mentioned in Subsection 10.2.2, correct, full treatment requires that simulation of production and decay of a particle are not separated. A proper design of the architecture should take into account this, physics motivated, constraint.

[5] i.e. in context of software for the Linear Collider

Chapter 11

Overview of event records

In this chapter I shall present the overview of the most important standards related to event records. The HEPEVT standard will be covered in most detailed way, as being most popular and most widely used so far; other standards, including the ones implemented in C++, will be mentioned and discussed as well. The HEPEventLib prototype of "event record unification layer", which I developed for the purpose of use in MC-TESTER (and, initially, in the C++ implementation of PHOTOS [56]) will also be presented, as a prototype of viable, iterator-based solution, used successfully in other programs.

11.1 HEPEVT Event Record

The HEPEVT standard for event record has been proposed in 1989, in [106]. The proposed standard contained the main `/HEPEVT/` common block, and the optional `/HEPSPN/` for spin information.

11.1.1 `/HEPEVT/` common block

The main commonblock of the standard, namely the `/HEPEVT/` common block, contains the information on particle content, event history and production vertexes. The structure of the `/HEPEVT/` common block is presented in figure 11.1.

According to the naming convention that was established, the names of all the variables in the specification contained the three characters `HEP` (for High Energy Physics), to make them distinguishable and to avoid overlaps with other names.

The original common block was able to store up to 2000 entries, however it was assumed that the `NMXHEP` parameter will be actively use, to ensure that the limit is never surpassed.

It was also decided that special initialisation and finalisation records may be encoded. They should be marked by negative value of the `NEVHEP` variable.

A record with `NEVHEP`=-1 , written e.g. by initialisation routine, would encode specific information in the first entry (`IHEP`=1) in the following order:

- `ISTHEP(1)`: user-defined run identifier
- `IDHEP(1)`: user-defined generator identifier
- `JMOHEP(1,1)`: date of run in `yymmdd` form
- `JMOHEP(2,1)`: time of run in `hhmmss` form
- `JDAHEP(1,1)`: version and subversion of generator in `vvss` form

Figure 11.1: The structure of /HEPEVT/ common block, as proposed in [106].

```
      parameter(NMXHEP=2000)
      common/HEPEVT/NEVHEP,NHEP,ISTHEP(NMXHEP),IDHEP(NMXHEP),
     &JMOHEP(2,NMXHEP),JDAHEP(2,NMXHEP),PHEP(5,NMXHEP),VHEP(4,NMXHEP)
```

The variables have the following meaning (IHEP: index of particle in the arrays):

- NMXHEP: defines maximum number of entries
- NEVHEP: event number
- NHEP: the actual number of entries stored in this event
- ISTHEP(IHEP): status code of entry IHEP

ISTHEP	meaning:
0	null entry
1	final-state particle which does not decay
2	particle which decayed, part of event history
3	documentation line (not event history), eg. two reacting particles, etc
4-10	reserved for future standards
11-200	for use by specific program
201-	for use by users, eg. event tracking in detector

- IDHEP(IHEP): particle identity (according to PDG specification)
- JMOHEP(1-2,IHEP): pointer to the position of first and second (optional) mother
- JDAHEP(1-2,IHEP):pointer to the position of first and last daughter
- PHEP(1-3,IHEP: particle momentum in x,y,z direction $[GeV/c]$
- PHEP(4,IHEP): particle energy $[GeV]$
- PHEP(5,IHEP): particle mass $[GeV/c^2]$, $-\sqrt{-m^2}$ for spacelike partons (negative mass)
- VHEP(1-4,IHEP): production vertex position $[mm]$, and time $[mm/c]$

- JDAHEP(2,1): last date of change of generator in yymmdd form

A record with NEVHEP=-2 , written e.g. by a routine called after all events were generated, would encode specific information in the first entry (IHEP=1) in the following order:

- ISTHEP(1): number of events generated in the run
- PHEP(1,1): total cross-section for the simulated processes obtained by Monte Carlo integration in [mb].

Special IDHEP identifiers were also defined for a cluster (91), a string (92), an independent fragmentation parton system (93), and a showering parton system (94). The flavour content of such object, as well as details of momentum sharing, had to be determined by looking at the mother partons, i.e. the two partons in positions JMOHEP(1,IHEP) and JMOHEP(2,IHEP) for a cluster or a shower system, and the range JMOHEP(1,IHEP)–JMOHEP(2,IHEP) for a string or an independent fragmentation parton system.

What concerns the event history, it was defined that the daughters of a particle indexed by IHEP are stored sequentially, in one, contiguous region JDAHEP(1,IHEP) – JDAHEP(2,IHEP); for cases when there is only one daughter (e.g. $K^0 \to K^0_S$), both indices should be defined, to make a uniform approach in terms of loop constructions.

11.1.2 /HEPSPN/ Spin information common block

The encoding of spin (polarisation) information is not a trivial task: there is no simple approach that would be also general enough to cover all eventualities. For the spin-$\frac{1}{2}$ particles, the spin polarisation density matrix may be encoded using the spin four-vector s. The values of the spin four-vector s are stored in the dedicated common block, which is parallel to /HEPEVT/, entry by entry:

```
COMMON /HEPSPN/SHEP(4,NMXHEP)
```

This common block needs to be accompanied by a prescription which describes the transformation properties of the spin four vector s when the particle four-momentum p is changed in boosts and rotations. In particular, if the set of transformations puts the fermion back to the rest frame (e.g. to generate decay), the quantisation axis for the spin vector needs to be cared of (Wigner rotation needs to be applied).

There was no restriction put on the modulus of s (in the rest frame) apart from the trivial one $|\overrightarrow{s}| \leq 1$. A pure spin state corresponds to $|\overrightarrow{s}| = 1$. Correlations between particles (e.g. in a $\tau^+\tau^-$ pair) might be simulated (in approximate way) by an appropriate choice of the spin vectors at the production vertex; different strategies might be possible, but needed not to be specified as part of the standard.

11.1.3 Update at LEP-2

In 1995, the need for the update of the existing structure of the HEPEVT event record was expressed at the meetings of the LEP2 QCD Event Generator Working Group [107]. An extension of the standard, which took into account the requirements for QCD-EW interfacing has been proposed [108], and generally accepted as the new HEPEVT standard. At the technical level, the /HEPEVT/ common block was modified.

The /HEPEVT/ common block was modified to accommodate up to 4000 particle (NMXHEP=4000), and all the type of all real variables was changed to DOUBLE

PRECISION. The new data structure became incompatible with the original one; hard-to-detect bugs started to appear when blocks of code with differing definition of the /HEPEVT/ common block were used together. One of the ways to address this problem will be presented in Chapter 16, where the TAUOLA-PHOTOS-F package is described.

At the physical level, a specification of the interface between QCD (Parton Shower) event generators and the four-fermion Electro-Weak (Matrix Element) event generators was proposed. The encoding of the colour flow information was achieved by defining a convention in which the fermions outgoing from the electroweak generator should be stored in the event record. Still, the proposed solution was not sufficient for the general case.

The new standardization have also proposed the new way to deal with particle codes (covered in section 11.2) and proposed the concept of common decay tables containing branching ratios and basic matrix elements for particle decays.

11.1.4 The use of HEPEVT

The HEPEVT specification (and the /HEPEVT/ common block) became de facto standard for data exchange between HEP simulation (and data analysis!) modules, from matrix-element to detector simulations. For more than a decade, this standard remains the only one that was generally accepted and followed to this date. In spite of its limitations, discussed already in the previous chapter, this standardisation (and its general adoption in the HEP community) was an important milestone; it enabled the possibility of creation of "afterburner" algorithms, such as PHOTOS and TAUOLA, that enrich the physics content of the events produced by the main Monte Carlo programmes.

The simulation modules use different approaches (and "philosophies") to treat the event record. Let me present a few examples here.

PHOTOS operates directly on the contents of HEPEVT: it extracts the information about the decay cascade, and modifies both: the structure of the cascade (added photons) and the properties of the particles (addition of photon requires re-calculation of four-momenta of all daughter particles in a tree). In the case of TAUOLA, a dedicated interfacing module, TAUOLA Universal Interface (see also Chapter 14), is used to extract the data from HEPEVT, and to append the products of the τ decays, generated by TAUOLA, at the end of HEPEVT. PYTHIA has its own standard for the event record (LUJETS/PYJETS), and provides functions to translate the data from/to HEPEVT; three options for the event record grammar are possible to specify [109] - they address the needs for various details of the event history information, expressed by experimental collaborations. In HERWIG, HEPEVT is used as internal representation of the event record: conventions were specified to be able to "force" additional information (i.e. colour flow) into the mother-daughter pointers, and the status code; as a result, the data stored in the /HEPEVT/ common block is difficult to interpret in other modules.

The evolution of the HEPEVT standard, and (physics model motivated) non-standard methods motivate the creation of diagnostic tools (or infrastructures), capable of detection of problems with incompatibility that occur when a module is installed in a large simulation chains. Being sensitive to consistency of the HEPEVT data, PHOTOS (and its event-record sanity-checking routines) was used many times as a debugging tool for large collaboration software chains. A tool such MC-TESTER (described in Chapter 17), that allows to perform semi-automatic tests of installation correctness is also highly demanded.

11.2 PDG and StdHep specification for particle numbering

In contract to the case well-defined physical properties, such as momenta, or vertex coordinates, where it is relatively clear to agree on the format for storing the data (i.e. agreeing on the use of natural units, where $c = 1$, the order of components in the momentum-energy four-vector), the convention for encoding the information about the type of the particle is far from trivial. For the simulations in late 1980's, a viable solution seemed to be to assign an integer number to any particle type, according to some convention (e.g.: 22 for photon, 11 for electron, -11 for positon, 13 for muon).

The numbering scheme for Monte Carlo simulations has been introduced in 1988 in [110]. It however turned out, that the scheme, which was limited to the set of known particle, was insufficient for studies of new (to be discovered) particles. To alleviate the situation, a new particle-id coding-scheme was proposed in the new standardisation [108]. Although it was not 100% compatible with the PDG numbering scheme at that time[111], it helped to establish a common standard and avoid ad-hoc solutions i.e. for the particles which were not yet covered in the PDG tables.

Since 1998 the new scheme has been adopted as a standard numbering scheme for the Monte Carlo generators[112]. The most recent version of the specification, at the time of this writing, is [113]; the most up-to-date version, including computer-readable tables, is maintained at the Particle Data Group web page[114]. The Particle Data Group maintains this numbering in cooperation with Monte Carlo authors and the StdHep Monte Carlo Standardisation project at Fermilab, as part of a system to facilitate translation between different Monte Carlo event generators and detector simulators. The StdHep lists contain a few specialised codes beginning with 99 which are needed for particular event generators, but which are not included in the list of PDG particle numbers.

11.3 StdHep: common output format for Monte Carlo events

The further standardisation of the event record, and extensions towards easier used in component-based generator systems was done in scope of the StdHep. The current version of the standard [115] consists of the revised version of the HEPEVT event record, a particle numbering scheme (subsection 11.2), the "Les Houches accord" extension for encoding the information required to interface parton-level generators with showering/hadronization generators, as described in chapter 11.4, and the extensions for multiple-interactions and general-information encoding, which are presented in Figure 11.2.

The StdHep standard also provides a library of functions to convert the data from internal event records of the ISAJET, PYTHIA, HERWIG and QQ generators to the StdHep data structures. There are also routines that transports the data from the StdHep data structures to the ones of the GEANT detector simulation package (KINE stack, vertexes positions recorded by calls to GSVERT; the GEANT particles in the stack are cross-referenced to the line numbers in StdHep through the user words in GSKINE). The library also contains routines to store/retrieve the event data in files, and an event display. Another integral part of the StdHep is a table of particle masses, as published by the Particle Data Group; the data in this table is populated by a call to PDGRDTB subroutine, which reads the content from the file indicated by the PDG_MASS_TBL environment variable.

Figure 11.2: Auxiliary common blocks defined in the StdHep standard

```
      integer numlti,jmulti
      common/hepev2/nmulti,jmulti(nmxhep)

      integer nmxmlt
      parameter (nmxmlt=16)
      integer nevmulti,itrkmulti,mltstr
      common/hepev3/nevmulti(nmxmlt),itrkmulti(nmxmlt),mltstr(nmxmlt)

      double precision eventweight, scalelh
      double precision alphaqedlh, alphaqcdlh, spinlh
      integer icolorflowlh, idruplh
      common/hepev4/eventweightlh,alphaqedlh,alphaqcdlh,scalelh(10),
     +              spinlh(3,nmxhep), icolorflow(2,nmxhep), idruplh

      double precision eventweightmulti, scalemulti
      double precision alphaqedmulti,alphaqcdmulti
      integer idrupmulti
      common/hepev5/eventweightmulti(nmxmlt),alphaqedmulti(nmxmlt),
     +              alphaqcdmulti(nmxmlt),scalemulti(10,nmxmlt),
     +              idrupmulti(nmxmlt)
```

HEPEV4: filled after generation is finished, not intended for PSG to MEG interface:

- `IDRUPLH`: identity of current process, as given by LPRUP codes
- `EVENTWEIGHTLH`: event weight $(\frac{total\ crosssection}{total\ generated})$
- `QEDLH`: QED coupling α_{em}
- `QCDLH`: QCD coupling α_s
- `SCALELH(10)`: squared scale Q of the event
- `SPINLH(3)`: spin information
- `ICOLORFLOWLH(2,...)`: (Anti-)colour flow

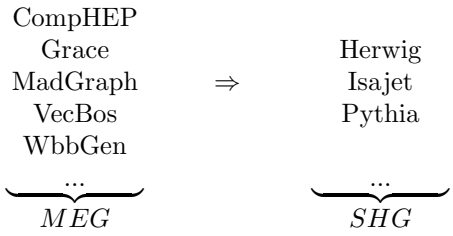
Other blocks: store the information about multiple interactions:

- `NMULTI`: number of interactions in the list
- `JMULTI(...)`: multiple interaction number
- `NEVMULTI(I)`: event number of original interaction
- `ITRKMULTI(I)`: first particle in the original interaction
- `MLSTR(I)`: input stream identifier
- `IDRUPMULTI(I)`: identity of the original interaction
- `EVENTWEIGHTMULTI(I)`: event weight of the original interaction
- `ALHAQEDMULTI(I)`: QED coupling of the original interaction
- `ALHAQCDMULTI(I)`: QCD coupling of the original interaction
- `SCALEMULTI(...,I)`: scales of the original interaction

11.4 Les Houches accord

Increasing complexity of calculations required by QCD, caused the appearance of a multitude of dedicated matrix-element hard-process event generators, operating at partonice level. The use of events produced by such generators were obviously limited to studies of theoretical models - partons are not direct physics observables, and need to "hadronize". The two most popular, non-perturbative, parton shower and hadronization models are the ones implemented by PYTHIA and HERWIG, yet due to limited flexibility of HEPEVT, there was no standard way to pass all adequate information from the matrix-element generator, to the parton shower/hadronization generator.

To accommodate this trend for modularity in Monte Carlo event generators, several authors of popular Monte Carlo programs attending the *Physics at TeV Colliders Workshop* in Les Houches, 2001, have agreed on a generic format for the transfer of parton level event configurations from matrix element generators (MEG) to showering and hadronization event generators (SHG) [116]. The "Les Houches Accord" is also described in [105].



Events generated this way are usually refereed to as "user-defined" processes, to distinguish them from internal processes that come with the SHG. The solution that was proposed: a FORTRAN common block structure for the user-process, was not intended to replace HEPEVT standard. The user-process common blocks address the communication between two generators only: a MEG one and a SHG one, and not the communication of event generators with the outside world (e.g with analysis/detector simulation software).

Two common blocks were defined: one to store the general parameters, concerning the whole run, to be exchanged by the generators at initialization, and one to exchange the information every time a new event is generated and transferred from MEG to SHG. The layouts of these common blocks are presented in figures 11.3 and 11.4. The specification suggests also the names for two subroutines, `UPINIT` and `UPEVNT`, that will link the two generators at initialisation and event generator stages respectively.

The user-process common blocks tries to resolve the most important deficiencies of HEPEVT standard - it defines the way to store colour-flow and spin information (in one of possible approximation models) . The proposal addresses also various technical issues, specific to Monte Carlo techniques, like different event weight model and treatment of subprocesses. It also clearly states that the meaning of the user-process common block must not be overloaded - all the extensions must be performed using other, dedicated, implementation-specific common block.

A C++ implementation of the "Les Houches accord", containing similar information content, was also proposed for the HepMC event record in Ref. [118].

Figure 11.3: Les Houches Accord: the layout of the initialisation common block.

```
      integer     MAXPUP
      parameter ( MAXPUP=100)
      integer     IDBMUP, PDFGUP, PDFSUP, IDWTUP, NPRUPM LPRUP
      double precision EBMUP, XSECUP, XERRUP, XMAXUP
      common /HEPRUP/ IDBMUP(2), EBMUP(2), PDFGUP(2), PDFSUP(2)
     +                IDWTUP, NPRUP, XSECUP(MAXPUP), XERRUP(MAXPUP),
     +                XMAXUP(MAXPUP), LPRUP(MAXPUP)
```

The variables in the common block have the following meaning:

- `MAXPUP`: defines maximum number of processes
- Beam information: beam particles - 1 and 2 - travel along $\pm Z$ direction
  - `IDBMUP(2)`: ID of beam particles (according to PDG (113))
  - `EBMUP(2)`: energy of beam particles $[GeV]$
  - `PDFGUP(2)`: PDF author group for beams, according to (117)
  - `PDFSUP(2)`: the PDF set ID, as above; for e^+e^- or when SHG defaults are used, `PDFGUP`=1-, `PDFSUP`=-1
- Process information:
  - `IDWTUP`: defines how the event weights (`XWGTUP`) are interpreted,

`IDWTUP`	event selection according to	control of mixing/unweighting	`XWGTUP` input	output
+1	`XMAXUP`	SHG	+weighted	+1
-1	`XMAXUP`	SHG	±weighted	±1
+2	`XSECUP`	SHG	+weighted	+1
-2	`XSECUP`	SHG	±weighted	±1
+3	-	user interface	+1	+1
-3	-	user interface	±1	±1
+4	-	user interface	+weighted	+weighted
-4	-	user interface	±weighted	±weighted

  - `NPRUP`: the number of subprocesses
  - `XSECUP(J)`: the cross-section for process J $[pb]$
  - `XERRUP(J)`: the statistical error of J'th cross-section $[pb]$ (for printouts)
  - `XMAXUP(J)`: the maximum `XWGTUP` for process J (mandatory for `IDWTUP`=$\pm 1, \pm 2$)
  - `LPRUP(J)`: the list of user process IDs for this run

Figure 11.4: Les Houches Accord: the layout of the event-data common block.

```
      integer     MAXNUP
      parameter ( MAXNUP=500)
      integer     NUP, IDPRUP, IDUP, ISTUP, MOTHUP, ICOLUP
      double precision XWGTUP, SCALUP, AQEDUP, AQCDUP,
     +                 PUP, VTIMUP, SPINUP
      common /HEPEUP/ NUP, IDPRUP, XWGTUP, SCALUP, AQEDUP, AQCDUP,
     +                IDUP(MAXNUP), ISTUP(MAXNUP), MOTHUP(2,MAXNUP)
     +                ICOLUP(2,MAXNUP), PUP(5,MAXNUP), VTIMUP(MAXNUP),
     +                SPINUP(MAXNUP)
```

- General:
  - `MAXNUP`: defines maximum number of particle entries
  - `NUP`: number of particles in this event
  - `IDPRUP`: ID of the process for this event
    when `IDWTUP`=$\pm 1, \pm 2$, process selected by SHG;
    when `IDWTUP`=$\pm 3, \pm 4$, process selected by MEG
  - `XWGTUP`: event weight; see also the description of `IDWTUP`
  - `SCALUP`: scale of the event $[GeV]$, as used for PDFs
  - `AQEDUP`: α_{QED} for this event (-1 if not specified)
  - `AQCDUP`: α_{QCD} for this event (-1 if not specified)
- ID, Status and Parent-Child History
  - `IDUP(I)`: PDG ID (113)(0 for non-physical)
  - `ISTUP`: status code:
    - -1: incoming particle
    - $+1$: outgoing final state particle
    - -2: intermediate space-like propagator, x and Q^2 preserved
    - $+2$: intermediate resonance, mass should be preserved
    - $+3$: intermediate resonance, for documentation only
    - -9: incoming beam particles at time $t = -\infty$
  - `MOTHUP`: index of first and last mother
- Colour Flow: tags for colour flow lines
  - `ICOLUP(1,I)`: passing through the colour of the particle
  - `ICOLUP(2,I)`: assing through the anti-colour of the particle
- Momentum and Position
  - `PUP(5,I)`: lab frame momentum (p_x, p_y, p_z, E, M) $[GeV]$.
  - `VTIMUP(I)`: invariant lifetime $c\tau$ $[mm]$
- Spin/Helicity
  - `SPINUP(I)`: cosine of the angle between the spin-vector of particle I and the momentum of decaying particle, specified in the lab frame; -9 for unpolarised/unknown.

11.5 Other efforts in FORTRAN-based generators

11.5.1 LUJETS/PYJETS

The most successfully alternative for HEPEVT was the event record used in the series of the Lund Monte Carlo event generators, namely JETSET and PYTHIA. The event record common block called LUJETS, later slightly modified and renamed to PYJETS (in PYTHIA 6) was based on the same concept that HEPEVT. The mechanism for compatibility with HEPEVT was implemented in form of a function (LUHEPC, later PYHEPC), that converted the data between the two event record data structures.

11.5.2 Super Monte Carlo System

In 1989–1990, a group of physicists from Monte Carlo Simulation Laboratory Group (MSL) working on Eloisatron project postulated the construction of SMCS: Super Monte Carlo System[53]. The main idea behind SMCS was to hide the complexity and technicalities from the user and provide unified view of the data used by subsequent modules of the whole simulation chain. The design directives of the project were: portability, transparency, user-friendliness, modularity and segmentation.

To provide transparent access to the data, the project proposed the Monte Carlo Event Generator Adaptor (MEGA) package[119]. The ideology of the package was close to the Entity-Relationship model, known widely from the model of relational databases. From technical point of view, a new event record standard, HEPEVE [120], which was an alternative to the HEPEVT standard, was proposed.

Ultimately, the COSMOS (COmprehensive Super MOnte carlo System) project [54]was intended to be an implementation of the idea of a Super Monte Carlo System. The project was however abandoned - the data structure was never used in any of mainstream generators; the remnants of the software-management system may however still be visible in some computer codes originating from that time, e.g. in PHOTOS.

11.5.3 Problems with scalability of Monte Carlo codes

At the time of Super Monte Carlo Systems, when plans for future Monte Carlo simulators for supercolliders were tailored, it started to be apparent that the programming technology used till then may not be sufficient anymore for the new, complex then never before systems [68]. Upon the practise from LEP, it was noticed that although the HEP experiments usually plan the structure of their software carefully at the beginning, the state of the software deteriorates quickly. Among the problems, the following seemed to be the most common:

- ❏ lack of re-utilisation of existing code
- ❏ need for maintenance of multiple, incompatible versions of the same code
- ❏ problems with interfacing the modules written by various groups of programmers
- ❏ the code being written "in hurry", with no documentation, coding conventions not being enforced

The limitations of the FORTRAN-based simulation codes became apparent as well. The most important issues were:

- ❏ lack of dynamic data structures, which needed to be compensated using external "memory management" packages, mainly ZEBRA [] and ADAMO []

- lack of flexibility in construction of complex data structures (no compound data types); in effect, any extension of common-block-based data structures usually made them incompatible with older versions, which made the issue of matching the versions of data types in subsequent MC modules quite complicated; the incompatibility of the HEPEVT data structure declaration was amongst the most common reasons of errors in large simulation chains.

- name space being to small: the names of the functions (initially limited to 7 characters) resulted in using cryptic acronyms as names of the symbols in the code; in large simulation systems, composed of a number of modules, eliminating possible conflicts in names was an issue. The "relaxed" syntax of new compilers allow for longer names (the convention used e.g. in KKMC)

The conjecture that applying Object-Oriented programming technologies and data abstraction may solve majority of these problems, as expressed e.g. in [68] resulted in C++ being accepted as the programming language for all experimental collaborations in the LHC.

11.6 C++ event records

11.6.1 The FORTRAN→C++ transition

The decision of using C++ and Object-Oriented programming approach in all experimental collaborations of the LHC brought new hopes and new worries to the area of HEP simulations. The naive understanding of Object Orientation expressed as simple encapsulation of functions with data structures was definitely one of the most dangerous assumption taken by many. In fact, the use of C++ was to be a revolution that brought the advance of more than twenty years of programming technology to the are of scientific software dominated by FORTRAN. The availability of compound data structures, the "hell and heaven" of pointers and dynamic data structures, advanced containers and iterators provided by standard libraries (STL), and the need to change the view on the organisation of the code from procedural to object-based was initially overwhelming to many authors of computer codes, accustomed to the simplicity and clarity of FORTRAN, as applied to scientific codes. The conflict between computer-science experts, fascinated with the new technologies and high abstraction levels, and pragmatic physics community was unavoidable... The clarity and compactness of FORTRAN codes confrontend with "bloat" of cryptic C++ language construct, such as templates, traits, persistency, object-oriented databases and fancy graphical tools with millions of options to master could not convince that transition to C++ could be profitable; the promise of "elegance" and "simplicity" given by Object-Oriented approach seemed to be far away.

11.6.2 Role of the ROOT package

Few were the projects (and the authors) that succeeded in bringing the two worlds to agreement at that time. The most prominent example is the ROOT project, which chose the "pragmatic C++ approach" to implement a new, object-oriented version of the CERN analysis software packages such as PAW, MINUIT, HBOOK and ZEBRA. The simplicity and clarity of the C++ construct, available through the command-line C++ interpreter (and simplified, C++-based macro language) convinced many users of the original, FORTRAN-based packages to invest effort in migration to the new tool. Resemblance of techniques adopted in ROOT to the techniques used in the original packages (C++ macros replacing KUMACS

macros, approach to histogramming, the immediate availability of all analysis-aiding packages related to data fitting) were definitely an important factor speeding-up the adoption of ROOT, and C++. The adoption of ROOT was almost immediate in HEP laboratories in the USA, for instance in the STAR experiment at BNL, which based its software architecture on ROOT completely. For the case of the LHC experimental collaborations, ROOT was initially treated with a lot of reserve; currently it has been adopted to smaller or later degree by all of them. Flexibility and suitability to the needs, matureness of approaches inherited to some degree from the time of LEP-era software packages, and very good performance and reliability of implemented solution cleared a path for ROOT to be used at least as object persistencency and data storage layer for the off-line analyses.

Among other projects that started to exploit C++ at early stage at CERN, was the new object-oriented implementation of the GEANT detector simulation package, GEANT4. Application of the classical software development process methodology was evaluated in that project. The scale of the project and the richness of its physical content resulted in problems with reliability of the results produced by the new models in GEANT4, and a long process of validation and reliability testing.

11.6.3 C++ event records: current situation overview

Despite over a decade that elapsed since the decision of use of C++ in the LHC experiments, a commonly agreed standard for C++ event record (of the power of HEPEVT) does not exist. A variety of proposals were made, mainly based on the heritage of FORTRAN event records, some of them being accepted by some experimental collaborations. None of the proposed standards was flexible enough, free of errors of its FORTRAN predecessors, and easy in use to become a generally favoured option. The two projects that are of the most significant nowadays are probably HepMC and ThePEG; the former being a format chosen for instance by the ATLAS collaboration, the latter being used to develop the new versions of HERWIG (and Pythia7 prototype) event generators.

The lack of a standard C++ event record is one of the most important reasons for existing simulation codes still being maintained in form of FORTRAN codes, in majority interfaced by means of the HEPEVT event record. From my own practise [56] I conclude that the sole process of translation of a code seems to be relatively easy (once the appropriate methodology and tools, such as MC-TESTER, are available), yet such effort is not justified if C++ event record standard is not available. It becomes clear nowadays that the LHC experimental collaborations will rely on FORTRAN generators at least for initial few years of data analysis, and HEPEVT will be used (at least internally) for event record data handling. It is thus essential that proper interfaces to HEPEVT are implemented for every collaboration's software chain. For instance, in the solution chosen by ATLAS, the event record data is permanently stored using HepMC standard, then usually translated to HEPEVT by routines dedicated for data input/output from a module (such as PHOTOS or TAUOLA). Assuring the reliability of such interfaces is typically a common effort of module authors and the developers of experimental collaboration software.

In the next subsections I shall present an overview of historical and present C++ event record standards. I shall summarise their strong and weak points as well.

11.6.4 MC++

Amongst the earliest implementations of C++-based event records was the MC++ proposal [121] of a "C++ interface to event generators". The interface was based on two container classes: MCparticle and MCvtx and a set of auxiliary classes to store general event information and static particle type information. An event was

represented as a linked chain of MCparticle and MCvtx objects, representing particles and vertexes (for the first time, a separation of particles from vertexes was proposed; the concept used currently in HepMC). MC++ had a number of interface classes to generators such as ISAJET, PYTHIA, HERWIG, using the StdHep common blocks. MC++ was developed as a part of the D0 collaboration software; currently the on-line documentation is not available, and the code does not seem to be maintained anymore.

11.6.5 StdHepC++

The StdHepC++ event record[122] was another early proposed standards for the event record in C++. Initially it resembled much the HEPEVT common block, with a set of translation routines, later it evolved to an independent prototype of event record data structure. For some time, StdHepC++ exploited the concept of a container based on a "dag" graph [123]. The StdHepC++ standard seemed to resolve a group of problems of the original HEPEVT related to iteration over the structure: the use of a dag assured that the graph structure does not contain loops; at the same time, the use of a dag put artificial constraints on the models of the data that could be stored, and the issue of encoding general relations and levels of event history remained open.

StdHepC++ was distributed as a part of the StdHep package as of StdHep version 4.09 (January 2000); initially it had a form of C++ bindings. Since version 5.01 (June 2002), it started to be distributed with CLHEP package, rather than FORTRAN implementation of StdHep. In May 2002, a revised version of the StdHepC++ was released with CLHEP 1.8; the CLHEP/StdHepC++ was renamed to CLHEP/StdHep. Since then, the project does not seem to be managed by the StdHep group (the code repository was untouched since then) - it slowly drifted towards the HepMC event record (described below), which fulfils the requirements of StdHep specification. The concept of DAG container seems to have disappeared: it's functionality was replaced by robust iterators and vertex-particle dichotomy of HepMC.

11.6.6 CLHEP

The CLHEP project [124], Class Library for High Energy Physics, is a HEP-specific set of base (foundation) and utility classes, proposed at CHEP92. The classes, are organised in the packages, dedicated to random number generation, physics vectors, geometry, linear algebra, particle data tables (PDG); the StdHep event record, described above, is also included as a package of CLHEP. Recently, HepMC event record (which initially used only 4-vector package of CLHEP) was adopted as StdHep's implementation of event record; it is now included and maintained in scope of the CLHEP project.

The weakest point of CLHEP is lack of documentation: sole availability of automatically generated reference lists of functions and list, without clear explanation of concepts, examples of use and general introduction is capable to discourage any use of it.

11.6.7 HepMC

Even though not being a generally agreed standard, the HepMC event record [118, 125] format was adopted by a few experimental collaborations (ATLAS, CMS) as the main event record data storage for Monte Carlo data.

HepMC is based upon a set of relatively simple (and documented [118]!) classes, that organise the particles in tree-like graphs, connected by vertexes (being separate

objects); dedicated iterators are also provided to navigate over the particles and vertexes. It allows to store the spin-density matrices, colour flow information and any number of Monte Carlo weights associated to any vertex; the state of random number generator may also be stored, and the information prescribed by "Les Houches Accord", related to interfacing of matrix-element and parton shower generators may be fitted as well. HepMC provides interface routines to exchange the data with the HEPEVT event record, and perform file-based input/output operations. Internally it uses CLHEP's implementation of physics vector classes to encode momenta and vertex information.

Due to assumption of a simplistic topology for the data model, HepMC is limited in encoding relationships (the only implemented ones are the ones mediated by the use of vertex objects) and levels of event history. Multiple-collision events are emulated by employing parallel events; in ATLAS a solution to encode event history relationships was worked out using a convention for "barcode" object identifier numbers.

11.6.8 ThePEG

The most complex concept of the event record available nowadays is the one implemented in the ThePEG framework [126]. ThePEG is a comprehensive C++ toolkit for construction of universal HEP event generators, used at present by HERWIG++[127] and PYTHIA 7 [128] prototypes, and it is not intended to play the role of the universal, event record. Instead, it provides an architecture and infrastructure that is typically used by universal "all-in-one" generators, based on the approaches of PYTHIA and HERWIG, where event generation is decomposed to steps: hard process, parton showering, hadronization, particle and resonance decays[1]. ThePEG aims at providing the complete infrastructure, including low-level services such as persistency, data storage or memory management. Even though the use of component approach and modularity is clearly visible, the architecture does not seem to allow for reuse or replacability at these lowest levels[2]. From the technical point of view, the use of advanced C++ language constructs (template traits, guarded pointers) is impressive; the ultimately high level of programming abstraction may pose problems, even for programmers with more than intermediate experience and knowledge of C++! The complexity of class library would require that a well-written general introductory and overview documentation (manual) is available, which unfortunately is not the case. Sole availability of automatically-generated class reference documentation, covering more than 700 classes [129], discourages from any real evaluation of ThePEG even more...

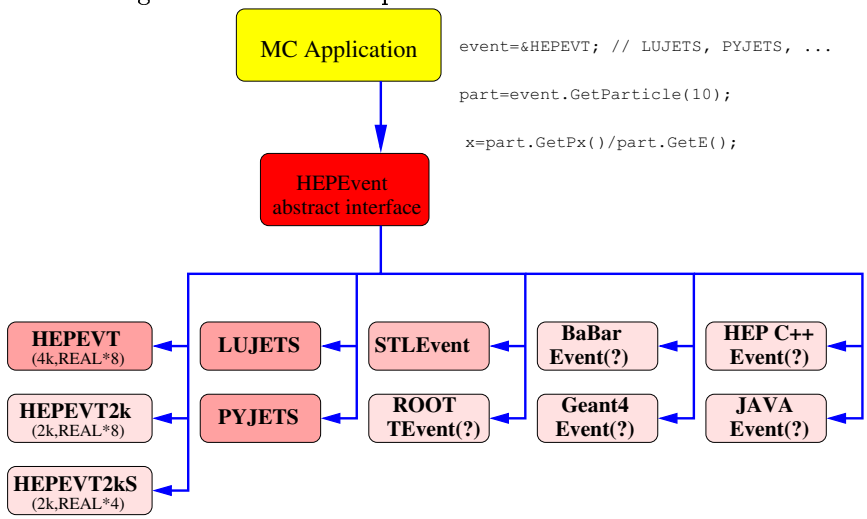
Despite the hardships related to the complexity (and lack of available documentation) of ThePEG, I would like to discuss its approach to the event record, as it brings the elements and solutions not existing before, and resolving a number of typical problems that made other event record structures fail.

The data model representing an event in ThePEG is based on a general graph, with non-constrained number of nodes and edges; the assumptions of the structure of the event are almost non-restrictive and in fact reflect the requirements presented in the previous chapter; for the first time, no tree, or dag, or ordered-list structure is assumed. Each event is composed of one or many collisions, and each collision is made of steps, which, in turn, are made of sub-processes. The objects representing particles enter the relations (encoded using pointers), such as decay cascade or

[1] This organisation of ThePEG limits its potential of use to a certain class of physics models; the simulations based on the full treatment of spin, such as KKMC or TAUOLA would probably be difficult to implement in this architecture.

[2] the use of alternative object-persistency mechanisms, such as the ones implemented in ROOT, or in GAUDI/ATHENA does not seem to be envisaged

Figure 11.5: The concept of the abstract event interface



colour flow, with objects representing steps and sub-processes. The instances of the same physical object in various steps are linked, and the navigation is possible by means of selector classes. Moreover, each particle has a link to a ParticleData object, containing the standard values of its physical properties. The links to process tables, matrix elements and other calculation model-specific objects are also maintained.

The event data model implemented in ThePEG is far too complex to be adopted as a general-purpose event record: the detail level of relationship encoding is too high in some parts, and assumes the use of a class of physical models; on the other hand, the handling of detector simulation or analysis is not envisaged at all. The rigidness of the architecture (imposed models for persistency, etc) and complexity of the language constructs are also factors preventing its potential adoption. Nevertheless, the design of the future event record could definitely be inspired by achievements and solution of ThePEG, at least in the event-generation part.

An approach similar to the one of ThePEG was taken by the authors of the SHERPA [130, 67] generator family, which uses its own architecture, and own data structures to internally store the event data representation. The multi-level event history structure is expressed by means of "blobs", which pass the information between the modules; classes implementing events, particles and vertexes are also available. SHERPA used to use PYTHIA to perform hadronization, hence a working interface to FORTRAN event record is maintained; file-based interfaces (i.e. to HEPEVT) and HepMC interface are available as well

11.6.9 HEPEventLib: unified access layer to event records

Already at the time of development of the `photos+`[56], a `C++` implementation of `PHOTOS` I turned my attention to the potential of use of C++ inheritance and polymorphism techniques to implement an abstract layer providing consistent methods of accessing event record data (from standard implementations of HEPEVT and LUJETS common blocks, at that time). Despite photos+ not ever being popularised and distributed wider, the significant amount of experience, and a proof-of-concept implementation of the event record abstraction layer turned out to bring profits later, in scope of the MC-TESTER project, the part of which is HEPEventLib - the abovementioned event record abstraction layer.

Is HEPEventLib yet another proposal for C++ event record standard? Defi-

nitely not. It is a demonstration of a flexible intermediate solution, for situation where no commonly-agreed standard exists, and data-exchange with various, incompatible with each other (i.e. at the level of programming language!) , standards needs to be provided. The solution allows to develop new algorithms using a clear and easy to use abstraction for the event record, and leave the details of accessing the actual data structure to the concrete implementations of the HEPEventLib interfacing functions (see Figure 11.5). Not only a unified, consistent view on event record data is provided to the programmer, but also the whole set of (supported by HEPEventLib) event record standards are immediately made accessible, including the ones that not yet exist at that time (for which the interfacing part of HEPEventLib may later be implemented). Hiding the technicalities of bridges between programming languages (C++/FORTRAN/Java/...), dealing with data overload or non-standard "flavours"[3] is natural in such approach, and separated from the development of the main code. Currently, the HEPEventLib layer is available for C++, and allows to access the data from FORTRAN event records (LUJETS, PYJETS, "standard" HEPEVT and HERWIG option for HEPEVT), yet potentially the same concept may be used to implement similar compatibility layers for other languages, including, for instance the access to the structures in Java from FORTRAN codes.

From the programmer's point of view HEPEventLib is composed of two parts: the interface specification (purely virtual abstract base classes are used) and a set of concrete implementation of this interface for various event record data structures. A simple implementation of physics vector classes (not compatible with CLHEP) is also provided. The UML diagram of HEPEventLib is specified in the Figure 11.6.

11.6.10 Other mechanisms

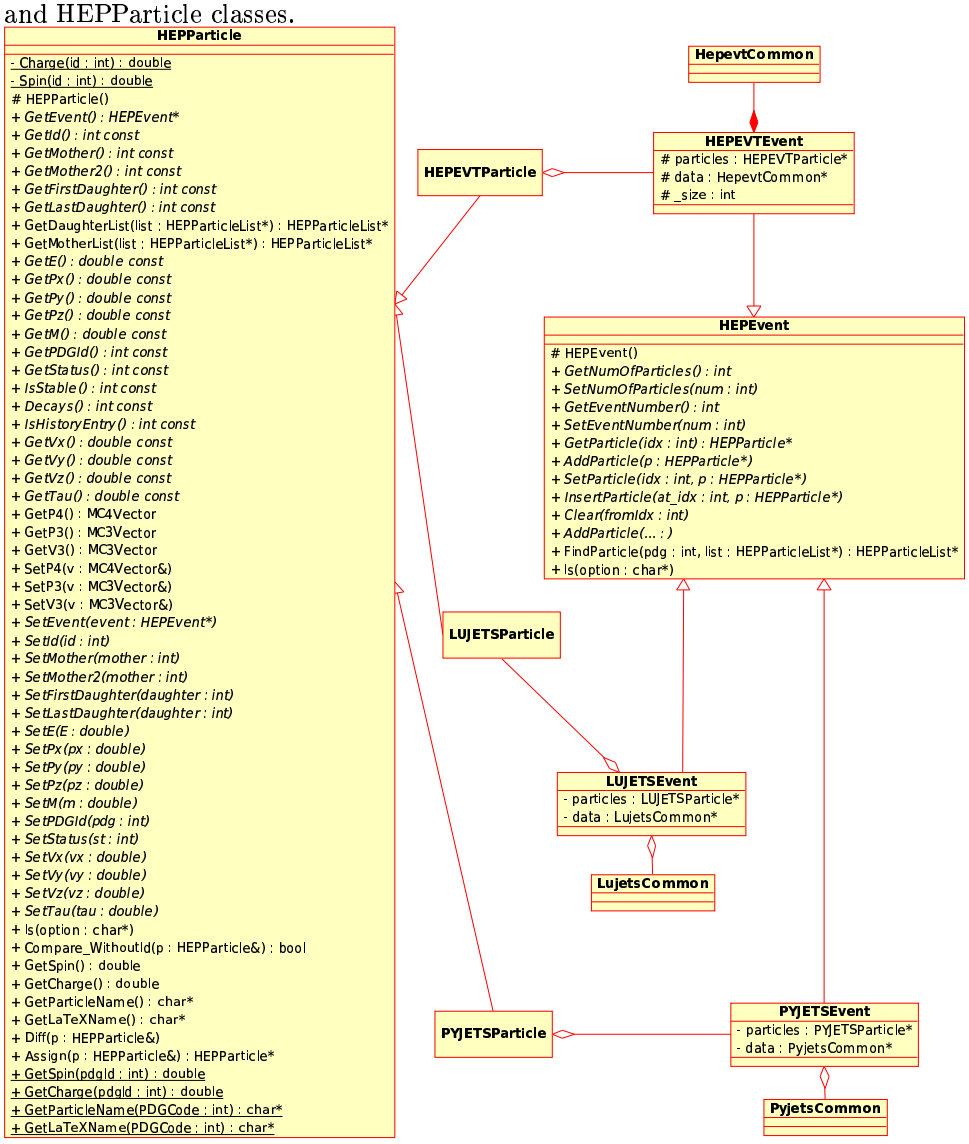
Lack of proper event record data structure often creates a need for ad-hoc mechanism for data exchange between simulation modules; a solution based on intermediate files storing the data are often the most easy to implement. For instance, in ATLAS off-line software the data is passed between event generation and subsequent detector simulation steps in form of ROOT files.

An interesting approach is to use native mechanism for inter-process communication provided by UNIX operating system, namely FIFO pipes, to transfer the data between two modules, out of which one is writing the data to the file, and the other is reading it, and the execution of the codes of the modules proceeds in parallel (synchronised, event by event though). Such approach was used for instance in Kandy [131], or in comparisons of 6-fermion generators performed in scope of the Linear Collider studies, described i.e. in [3], performed by means of MC-TESTER. In the latter case, the use of file-based approach allowed to eliminate the problem with different programming languages for the modules involved (FORTRAN77, FORTRAN90 and C++).

Despite the file-based solution may seem attractive because of the simplicity of its implementation and application, it should rather be used only in the cases where it is desirable to obtain some preliminary results, and the use of quick ad-hoc solution is generally justified. For more serious purposes, a solution based on proper interface implementation (such as the one of HEPEventLib) would be favourable; the technicalities of implementing FORTRAN-C++ interfaces (and example of translation of physics code from FORTRAN to C++) are described e.g. in [56].

[3] In the current version of HEPEventLib distributed with MC-TESTER, `the HEPEVT` common block is assumed to contain the matrices for 4000 `DOUBLE PRECISION` entries. To tune the size or precision of the `HEPEVT` (or `LUJETS/PYJETS`) to match your code, see `README` in the `include/` directory of MC-TESTER.

Figure 11.6: HEPEventLib UML diagram; the interface is specified by HEPEvent and HEPParticle classes.



11.7 Summary

Event record is a structure that allows to exchange physics information concerning events between modules in high-energy physics simulations. Complexity and multi-stage nature of the simulations, and event-generation in particular, favours the approach based on central event record data structure, and events being built and processed in stages, by subsequent modules. HEPEVT event record standard, used extensively throughout the LEP era proved the viability of this approach and allowed for modularisation of simulation codes. Nevertheless, problems with scalability and deficiencies of the FORTRAN language made HEPEVT standard diverge to various non-standard options, which was motivated by the increasing needs of physics models to fit additional information in the data structure that was too rigid. Despite C++ being in use (in the field of high-energy physics) for more that a decade now, a commonly agreed standard for event record is still not available. The existing proposals, such as HepMC, are based on the heritage of the HEPEVT, and due to artificial assumptions of the topology of physics process put too much constraints and are deemed to the same pitfalls that HEPEVT got caught into. On the other hand, large and comprehensive frameworks, such as ThePEG, for construction of event generators are created. Their data model for event description is well suited to describe event-generation. Nevertheless, they do not even try to fulfil the needs of later stages of simulations, such as detector simulation, reconstruction or analysis; they are not designed as event records. The richness of ideas contained there should, however, be an inspiration for the specification of the event record.

The LHC experimental collaboration adopted different solutions for event data storage: e.g. ATLAS and LHC have chosen to use HepMC. Nevertheless, at least for initial physics analyses, only the generators known to give reliable predictions, such as PYTHIA, implemented in FORTRAN will be used. A significant number of the new event generators, such as ALPGEN [132], or AcerMC [133]is implemented in FORTRAN and relies on the availability of FORTRAN-based event records.

This requires that interfaces are written to access FORTRAN event record data structures (HEPEVT, PYJETS) and translate them to C++ event records.

In the light of component-based approach to simulations, the validity of the HEP simulation architectures based on central event-record data structure is questionable. Physics arguments (requirements for full treatment of spin) also seem to support the conjecture that the methodology should probably be changed, to be able to describe wider class of physics models, and to take advantage of the component-based software technologies. The role of the event record data structure should probably be played by an interface specification. The considerations related to the physics-motivated requirements, presented in this part, remain valid for both approaches.

Part IV

Algorithms

Chapter 12

Overview of event generators

> **Disclaimer:** *the description of event generation process included in this chapter have been intentionally oversimplified, to make it understandable for a reader with relatively modest background in theoretical physics. A more detailed descriptions, covering the details of all presented aspects are widely available, yet the information provide may be difficult to understand and synthesize into a complete picture due to the multitude of concepts of theoretical physics invloved. Therefore the content of this chapter should only be treated as an initial overview, displaying the basic mechanisms and the meaning of subsequent steps of event generation, and motivating the reader to refer to a more complete and precise documentation.*

Event generators are the key component of the large software chains of the experimental collaborations, allowing to express the predictions of the models from theoretical physics in terms of objects which are the subject of study for experimental physicists: particles and resonances, together with their properties. The generators are therefore essential for making the *"theory=experiment"* equation, postulated at the very beginning of this thesis, hold in the physics analyses of high energy physics. A complicated process of confronting experimental data with the predictions of complete simulations, as presented in Figure 3.1 need to be completed.

In this chapter let me present an overview and classification of the event generators, and describe the actual multi-stage process of generating the event, which is typically performed collectively by a number of event generators, each of which dedicated to a particular task. The description presented in this chapter is not intended to be neither complete nor exhaustive - this topic is covered in a large number of publications and these; a detailed descriptions of the organisation of a typical, universal Monte Carlo event generator may be found, for instance, in [134], a quite complete and up-to-date presentation of the state-of-the-art in event generators may be found in [105]; another, slightly out-of-date, source of information, presenting the state of event generators at the era of LEP experiments is [135].

Event generators are Monte Carlo simulation programs (or sets libraries that need to be assembled to form a program) which, based on models of theoretical physics, produce events: lists of particles being the result of physics processes that occur upon a collision of high energy particles. Typically, the simulation of the complete event is a multi-stage process, performed collaboratively by a set of programs, each of which modelling a subset of physics processes to which it is dedicated; the programs communicate and exchange the data by means of the event record, described thoroughly in the previous part of this thesis.

Following the main concept of Monte Carlo simulations the generators produce

long series of events, which are subject of statistical analysis. To make the events produced by generators easily applicable as input to detector simulation a common practise is to generate the weight=1 events, which can then be given direct phenomenological interpretation.

"Exact" and approximated methods

The theoretical calculations for the high energy physics involve the use of theoretical models based, among others, on the Quantum Field Theory, and the Group Theory, used to build two major, well-established theories: Quantum Electrodynamics and Quantum Chromodynamics, and associated sets of theoretical models predicting the "new physics". Among the methods employed in the calculations, the most important are based on perturbative expansions, which allows to express the formulae for quantities such as cross-section as the infinite sum of terms, that converge (as more and more terms are taken into account) to the exact values and expressions. This way, asymptotically, any formula can be approximated to any arbitrary precision by including the subsequent terms of the expansion in the calculations. Typically, the expansion is the only way to express the (approximated) formulas in quantum mechanics: the simple analytical forms exist very rarely. In the quantum field theories, the expansion is usually performed in the powers of the coupling constants, which define the strength of the interaction. Therefore, one usually perform the theoretical calculations up to a certain, fixed order N, symbolically written as $O(\alpha^N)$, which means that only the terms which give the contributions that have the power of the α not larger that N are taken into account (e.g., calculation up to $O(\alpha_{QED}^2)$ will take into account the contributions of type $A_0 + A_1\alpha + A_2\alpha^2$, of the complete expansion $A = A_0 + A_1\alpha + A_2\alpha^2 + ... + A_N\alpha^N + ...$. Such approach is, however, effective only for the expansion that converge fast enough, that is for small values of α. In the case of QED, the $\alpha_{QED} \simeq \frac{1}{137}$ is small enough, so taking only the zero-th term ("Leading Order") already give a good approximation; in practise, it is not feasible to go to the order higher that 3 in QED calculations.

Systematic calculations performed order-by-order are however not always the best approach. Instead of summing all contributions of order up to N, a simpler formula (and better convergence) may be obtained by taking only the leading terms, yet up to some higher order M. For certain formulae it is even possible to extract the terms up to infinite order, and then sum them up into an analytical expression. The most popular approximation of that type is the "leading logarithms" approximation.

In the field theory, including subsequent terms of expansion is equivalent to including the Feynman diagrammes with more and more number of vertexes into the calculations of spin amplitudes. The number of diagrammes to be included grows very quickly with the order. On the other side, relatively big value of the strong coupling constant α_{QCD} would require considering higher-order terms. Moreover, the formulae for the Matrix Element are strongly enhanced whenever a gluon emitted, in e.g. $g \to q\bar{q}g$ process, is collinear to one of the quarks - the fact known as collinear enhancement. To resolve the problems of the soft and collinear divergences, which give dominant contributions to the Matrix Element a type of leading-logarithmic approximation may be used - the parton showers. In a simplified description, the generation of the events with the mechanism of parton shower is the following: at first, a signal "hard" interaction process is generated: typically it is some $2 \to n$ process with two incoming and n outgoing partons (where n is usually of order of a few). Usually this underlying process may be calculated with limited effort using Feynman diagrams. This hard process is also often referred to as a "signal process", i.e. it will describe the process that is of particular interest for performed study, hence it may be one of the well-known process from the Standard Model, but it may also include the processes predicted by new models, such as supersymmetric parti-

cles production and decay, etc. Then, the QCD content of the "signal process" is enriched by additional partons, that are generated from the incoming and outgoing parton lines. The partons that are generated may in turn emit subsequent partons, up to some limit, established by the energy scale of the event. Due to correlation effects, the new partons are generated in "cones" around the ones that emit them, which leads to shower-like, usually well-separated cascades of partons. In effect, after the parton-shower, an event may be built of many quarks and gluons (and eventually also other particles, like leptons, produced in the hard process). Such event is ready for the hadronization, i.e. grouping the quarks to form hadrons. Although the general mechanism for parton shower is unique, various model exist: they differ mainly in the criteria that determines when the showering process should be terminated (i.e. the termination variable, $t(\sim p_t^2)$ for PYTHIA is $t \sim Q^2$ (virtuality) and for HERWIG $t \sim (1 - \cos(\theta)$ (opening angle)).

MEGs versus PEGs

Because of the approach taken to perform the calculations, event generators for QCD may be classified into two categories: Matrix-Element generators (MEG) and Parton-Shower generators (PSG).

Matrix-Element generators are based on tree-level, complex calculation, which however lead to "exact" , systematic expansion in α_{QCD}. It is a very flexible method for Born-level multi-parton calculations, however the loop calculations are very difficult. This approach, however, give negative cross-secion in collinear regions which lead to unpredictable jet and event structure. They are also hard to interface with hadronization generators - the aspect is being currently studied intensively. MEG perform calculations on the parton-level, hence they cannot easily be confronted with the experimental results, which observe hadrons.

The parton-shower generators, on the other hand, are based on the leading-log approximation (or Next-to-Leading-Log). They provide a generic approach, independent on underlying "signal" process, hence their main topology cannot be pre-determined. As they rely on Sudakov form factors and resummation they give a sensible jet and event structure, which is easy to match to hadronization.

Hadronization

Whether the event was produced by a MEG or a PSG, the event produced by them (in perturbative process) will contain mainly quarks and gluons, accompanied eventually by a few leptons that might have been produced in the hard process. In the experiments, however, one observes mesons, baryons, leptons and photons, not the "bare" partons. The quarks and gluons need to be "hadronized", to form colour-singlet particles.

Hadronization is a non-perturbative process, hence there is no unique prescription for the hadronization: a few phenomenological models, which differ quite significantly, exist. The two most popular models are the string-fragmentation model [136], as implemented in PYTHIA [34], and the cluster-hadronization model [137], as implemented in HERWIG [33]. A modified cluster-hadronization model [138] is used in the SHERPA [130] generator.

Although the models differ in their approach, they give sensible results, which were meticulously compared with experimental data. There is no leading solution, hence the cross-check between the results from various models should usually be performed.

In the cluster-hadronization model, the gluons needs to be splitted to quark–antiquark (or diquark–antidiquark) pairs at first. Again, this process is based on a

parameterised, phenomenological model. Then, the quarks are grouped in colour-singlet clusters. The clusters may undergo "cluster fission" process, down to a certain cluster mass limit; after the threshold is passed, the quarks are converted to mesons and baryons of appropriate flavour.

Decays of heavy flavours

In the Standard Model, the *top* quark and the τ lepton are the only particles (elementary fields) that decay before hadronization (in the supersymmetric models the actual number of such particles is significantly larger). These may therefore be a source of additional parton showers. The particles in question are often fermions, hence the distributions of their decay products are affected by spin correlations between the production and decay of the fermion. In the majority of the Monte Carlo event generators these correlations are neglected, because in the Standard Model they are applicable only to the *top* quark and the τ lepton, however in the model of new physics, for instance supersymmetry, such particles are present and should be taken into account in an appropriate way. For the discussion of the spin in context of the event generation, see also Chapter 11.

The fact that the decays of un-hadronized particles may occur may also be important for the "after-burner" packages, like PHOTOS, which will be discussed in Chapter 13. Traditionally, PHOTOS is used to generated QED radiative corrections in the decays of particles and resonances, i.e. after hadronization. However, PHOTOS may also be applicable to the processes like the e.g. t-quark decays.

Decays of unstable particles

After the hadronization step, the colourless particles and resonances represent the state of the event (at this stage of event history). Only a small fraction of these particles are stable (i.e. does not decay in the volume representing the detector, which means certain relatively long-lived, unstable particles, like muons, are accounted here). The majority decay into lighter particles. The decay process is usually governed by the available phase-space, and the particle properties like mass and life time. Certain particles may also decay in more than one decay channel, with associated partial width (or branching fraction). Some particles, like the τ lepton, or B-mesons may decay via many decay channels, which are often not yet measured with a high precision. For such particles dedicated decay packages , like TAUOLA (described in Chapter 14) or QQ[139] were created.

Beam remnants, pile-up, minimum-bias and diffractive events, underlying event model. In the generation of events for purposes of detector studies, it becomes important to properly model not only the "signal" process, but also the aspects that - from theoretical point of view - seem to be unimportant, not interesting and possibly not easily calculable: the events produced by the generator should resemble - as much as possible - the results of real collisions of particles. In particular, the "beam remnants", that is partons that did not take part in the hard process, also hadronize and create hadronic jets at small scattering angles. Such jets or other relatively "soft" processes such as diffractive scattering, may not be interesting from the point of view of "discovery" experiments, yet proper modelling of them may be essential to correctly simulate the response of the detector, or design the systems for detector monitoring/calibration (e.g. luminosity monitor/detector).

The majority of hadron-hadron scattering events are typically not the ones that contain, say, Higgs production, but rather "uninteresting" ones, such as diffractive or elastic ones. Such events with low transverse energy, often refereed to as "minimum bias" events, are however important to estimate backgrounds or predict radiation

damage to the detector caused by scattered protons. The minimum bias with beam remnants are often considered as a part of the underlying event model. More discussions of this topic may be found i.e. in [140, 141, 142]. The role of separating the minimum bias events from "signal" events is dedicated to the trigger system of each detector.

At the LHC, each collision of the proton bunch will produce on average 23 individual proton-proton interaction. Proper modeling of the "pile-up" effect is a separate problem: the event generators typically produce the events being the result of single proton-proton interaction. Superimposing such events in the event record structures, which were not designed to hold the information of such type, becomes not only a technical but also a physical problem.

Universal versus dedicated Event Generators

The generators may also be classified as "universal" and "dedicated": the former, such as HERWIG or PYTHIA, offer the possibility of generating events with objects that are observed in experiments, i.e. they usually include a library of signal processes, parton-shower and hadronization routines and a library for decays of resonances. This kind of generator is particularly suitable for performing studies of the discovery potential for detectors, and testing of the new, often exotic models. This demands, usually, that significant number of approximations is used (such as not treating the spin information, or treating it in approximated way, such as narrow-width approximation) , and the precision is usually limited due to calculations being performed in the lowest order of expansion.

On the contrary, dedicated event generators, such as KORALZ, KKMC, WINHAC or TAUOLA are focused on high-precision measurements. They are usually developed in collaboration with experiments, and often incorporate the direct or indirect results of measurements and models tested by experimental collaborations. They typically perform calculations for a small set of related physical processes with higher-order corrections and employ systematic treatment of effects such as radiative corrections or spin correlations. The dedicated generators often become important elements of data analysis packages for the experimental collaborations. In fact, experimental collaborations usually maintain their own "collaboration software", with dedicated Monte Carlo generators, which fulfils the needs of precision in simulation of the physical event, the background, and detector response.

The list of the types of event generators, that collaborate during the event construction, presented above is not definite. In particular, there is a place for "content-enhancing" or "afterburner" generators, an example of which is PHOTOS, which complements an event with QED Radiative Corrections in form of photons being added in particle and resonance decays. Because of its atypical nature, it is particularly interested object for studies presented here, because it connects the realms of experimental and theoretical physics, combines various techniques while being compact, universal and modular. The next chapter is therefore dedicated to PHOTOS.

Chapter 13

PHOTOS: phenomenology and algorithm

A well-defined class of theoretical effects predicted by Quantum Electrodynamics consists of QED radiative corrections, which, at the ground of phenomenology, are often refereed to as bremsstrahlung: the emission of the electromagnetic radiation by the acceleration of charged particles. On the theoretical side, radiative corrections are the corrections to the predictions in high-energy physics which are due to Feynman diagrams containing additionally emitted photons (QED) or gluons (QCD). The additional diagrams may contain loops made of virtual particles (that lead to effects such as vacuum polarisation) or describe bremsstrahlung-like emission of real quanta. The effects due to (QED) radiative corrections typically affect the formulae for partial widths or cross-sections to a small degree, nevertheless they may turn out to be significant source of systematic error of theoretical predictions in high-precision measurements, such as the ones aiming at measurement of the CKM angles (see, for instance, [69]).

PHOTOS is a universal Monte Carlo algorithm that simulates the effects of QED radiative corrections in decays of particles and resonances. It is a project with a rather long history: the first version was released in 1991 [143], followed by version 2.0 [144] in 1994 (double emission, threshold terms for fermions). The package is in wide use [105]; recently it was applied as a precision simulation tool for W mass measurement at the Tevatron [145] and LEP [146, 147], and for CKM matrix measurements in decays of K and B resonances (NA48 [148], KTeV[149] , Belle [150], BaBar [151] and in Fermilab [152]).

Throughout the years the core algorithm for the generation of $O(\alpha)$ corrections did not change much; however, its precision, applicability to various processes, and numerical stability improved significantly. Increased interest in the algorithm expressed by experimental collaborations (including future LHC experiments) was a motivation to perform a more detailed study of the potential and precision of the PHOTOS algorithm; also new functionalities, such as multiple photon radiation and better interference corrections, were recently introduced. In this chapter we present the results of precision tests of the algorithm, documented also in [153]. The study of the precision of PHOTOS is presented in the main topic of the Chapter 18 of this thesis.

My personal involvement in PHOTOS dates back to 1999, where the C++ implementation of the algorithm was performed in scope of my MSc thesis [56]. Since that time I have been collaborating actively in the development and tests of PHOTOS [154, 109, 5, 153, 155].

13.1 Physics nature of the problem

Although QED bremsstrahlung in particle decays is one of the most elementary effects in quantum mechanics, it is not usually considered explicitly; to simplify, calculations are performed for inclusive quantities. In fact, it is only the case of a few specific decay channels, where exact fixed-order (e.g. $O(\alpha^2)$) fully-differential formulae, with spin amplitudes or matrix elements squared, are available in analytical, semi-analytical or Monte-Carlo form. Nonetheless, in the analysis of the experimental data, radiative corrections are usually treated together with the detector effects (e.g. conversion, detector efficiency) to form the "QED-subtracted" data. The control of the uncertainties of such data becomes a weak point: theoretical uncertainties appear on both sides of the $theory = experiment$ equation [156]. This problem does not seem to be so evident for "discovery" experiments or measurements performed on small statistical samples: usually the effects of radiative corrections do not exceed a few per cent. An exception is radiative corrections for background processes, such as $gg \to t\bar{t};\ t \to l\nu_l b$, where QED bremsstrahlung is an essential element for this channel background contribution to Higgs boson searches at the LHC in the $H \to \gamma\gamma$ channel [157, 158]. Conversely, for high-precision measurements (as those performed nowadays), good control of the radiative corrections becomes vital not only for the assessment of the overall experimental error of the respective cross-sections or branching ratios, but also for the shapes of the distributions.

The effects of radiative corrections gradually became an important topic in the context of such measurements as high-precision measurements of W-boson properties (see, for instance, the combined results of Tevatron runs [145]), or B, D, K meson decays for measurements of CKM-matrix coefficients in B-physics [159]. With increasing statistics of available experimental data, QED radiative corrections have become a significant element in the systematic error of the measured quantities.

Strict, systematic calculations performed order-by-order in the perturbation theory are usually not the most efficient way of including the effects of bremsstrahlung. To improve the convergence of the perturbative expansion, the most popular method in QED is exponentiation, a rigorous scheme of reshuffling the dominant terms between orders of expansion. This method is useful for the construction of Monte Carlo algorithms as well [55, 160]. In the leading-log approximation, partially inclusive formulae exhibit factorisation properties of QED, see e.g. [161]. A matrix element formula for particle decay accompanied by bremsstrahlung photon emission may be factorised to Born-level terms times the bremsstrahlung factor[1].

Similarly, a fully differential formula for Lorentz-invariant phase space for particle decay accompanied by a number of photons may be expressed in a way that exhibits factorisation properties, used, for example, in the construction of FOWL generator[25] or the TAUOLA algorithm for $l\nu_\tau\bar{\nu}_l(\gamma)$ [29]. The important property of this parametrisation is the full coverage of the phase space and exact (i.e. free of approximations) treatment. The price to pay is that the variables describing each added photon are defined in individual rest frames, separated by boosts, making the question of the choice of gauge fixing a very subtle point of the theoretical bases of the algorithm [164].

It is nonetheless straightforward to take advantage of these factorisation properties and approximate the fully differential formula for the cross-section in any particle-decay process accompanied by a bremsstrahlung photon, by a product of a cross-section formula that does not contain QED radiative corrections times a bremsstrahlung factor. The bremsstrahlung factor depends only on the four-

[1] For some cases this factorisation property is also present for non-approximated formulae, e.g. for the $O(\alpha)$ ME formula for Z-boson decays as implemented in MUSTRAAL [162, 163] at fully differential level. In [143] it was shown, contrary to the previous expectations, that this factorisation has a natural geometrical interpretation as well.

momenta of those particles taking part in the decay, and not on the actual underlying process! This approximation, which takes into account both real and virtual corrections, converges to an exact expression in the soft-photon region of phase space; this fact is exploited in the construction of PHOTOS. It exposes well-known infrared and collinear singularities. In the final formula, the infrared divergences that originate from expressions describing the emission of real and virtual photons may be regularised and cancelled out order-by-order. To realise this operation, a technical parameter giving the minimum photon energy is defined; then, integration over the directions of photons with energies lower than the cut-off is performed; this results in the final formula being free from infrared regulator. Only photons of energies higher than the cut-off are explicitly generated[2]. The PHOTOS algorithm properly treats the collinear region of the phase space as well: the singularities are regulated simply by the masses of the charged particles.

As a result of the studies of the second-order matrix element, in particular for $Z \to \mu^+\mu^-\gamma\gamma$ and $gg \to t\bar{t}\gamma\gamma$ (performed already in 1994 [157, 158]), the iterative properties of the bremsstrahlung-adding formula have been identified and a universal photon-emission kernel has been isolated and double-photon emission implemented in PHOTOS. However, at that time, the question of precision was left out of the discussion. The main application was an estimate of the background to the Higgs boson searches in the $H \to \gamma\gamma$ channel. Only double hard-photon bremsstrahlung was of interest, and a precision of about 10% on the cross-section was sufficient.

A flexible organisation of iteration and phase-space variables in the PHOTOS algorithm not only allows a full phase space coverage, but also seems to introduce some higher-order effects, as will be discussed later in the paper (subsection 18.2.2). This was achieved by careful studies and comparisons with matrix-element calculations [157, 158, 164], without need of any kind of phase-space ordering.

Some of the aspects of the construction of the single- and multiple-photon PHOTOS algorithms have been recently covered in our two articles: [155] (expands on the theoretical formulae behind the single-photon algorithm and the extension to include full first-order ME for leptonic Z boson decays), and [5] (describes the construction of the single- and multiple-photon algorithms expressed in terms of operators).

13.2 Evolution of the main features of the PHOTOS algorithm

PHOTOS is an "after-burner" algorithm, which adds bremsstrahlung photons to already existing events, filled in by a host generator (which does not take into account the effects of QED radiative corrections) and transmitted by means of the standard HEPEVT event record (only the information about four-vectors of particles taking part in the process, and the topology of the process are needed). PHOTOS adds (with a certain probability) final-state QED radiation in "any" decay of particle or resonance, independent of the physics process that was generated. As the result of its execution, bremsstrahlung photons are added in a fraction of the events in the HEPEVT event record, taking into account event topology and momentum conservation. The single-photon version of the PHOTOS algorithm originates from the MUSTRAAL Monte Carlo [162, 163], in its part for the $Z \to \mu^+\mu^-(\gamma)$ process. In PHOTOS the full $O(\alpha)$ matrix-element algorithm was simplified so as to isolate the universal kernel responsible for photon emission. The original algorithm, although

[2]From a practical point of view this poses no problem: the energetic resolution of calorimeters used in the experiments is also limited.

maintaining full $O(\alpha)$ precision and exposing factorisation properties, was however dependent on the underlying process[3]. A downgrade was therefore required to make the algorithm process-independent: the interference terms were removed from the formula, and later restored in an approximated way for a limited set of processes (decays of neutral particles into two charged particles of the same mass) by introducing a Monte Carlo interference weight. The first version of PHOTOS: universal Monte Carlo for QED radiative corrections in particle decays was released in 1991 [143]. The algorithm operated on the four-vectors stored, in the HEPEVT event record [166], by a host generator.

In 1994, version 2.0 of PHOTOS was released [144]. The most important improvement was an implementation of the double-photon emission obtained by iteration of the bremsstrahlung kernel. This version became widely used in the HEP experimental era [105]. In 2000, PHOTOS was integrated to the TAUOLA-PHOTOS-F environment (described also in Chapter 16) to ease code maintaining; the update of PHOTOS was released as part of the documentation of that package [154]. In 2003, driven by the needs of experimental collaborations, a dedicated Monte Carlo weight for $W \to l\nu(\gamma)$ process was developed [167]. This new option, implemented in PHOTOS version 2.07, was documented in [109]. One of the changes performed at that time, trivial from the physical point of view, but turned out recently to be vital, was the change of variables in the calculation of interference weight in PHOTOS: from angles to four-vectors.

In July 2004, we started a systematic study of the PHOTOS algorithm, focusing on its precision and possible extension to the multiple-photon emission. It turned out that the algorithm stability suffered from a bug; this had no impact on the results, but prevented the photon emission kernel to be iterated more that twice. Once the bug had been identified and corrected, the algorithm was extended to triple- and quartic-photon emission, then an exponentiated version of an iteration routine was implemented (see also subsection 13.4). In January 2005, driven by needs of B-physics experiments, a new, universal interference weight was implemented, allowing for the calculation of an interference for "any" process (see also subsection 13.5). In Winter 2005/2006 the full first-order matrix element for the leptonic Z decays has been installed as a proof-of-concept experiment, demonstrating the stability and flexibility of the algorithm [155]. The improvement in precision, introduced by means of a correcting weight, even though very interesting from the point of view of the study of the theoretical approximations used in the algorithm, are however too small to be noticed by any experimental collaboration, therefore this correcting weight is not likely to be included in the public version of PHOTOS.

13.3 PHOTOS stability improvements

Problems with numerical stability may occur when the four-vectors passed in the event record have insufficient precision. PHOTOS is very sensitive to rounding errors and momentum conservation in the event record, especially when the multiple-photon-emission mode of operation is used. At each iteration of the photon-emission kernel numerical rounding errors are accumulated. The value of the infrared cut-off parameter was raised to protect the algorithm from being stopped because of numerical instability. It is not the desired method to solve the numerical problems of course...

[3]It remains implemented in the first published version of KORALZ 3.8 [165], with multiple initial-state bremsstrahlung and single final-state radiation. Technical organisation is, nevertheless, different.

13.3.1 Subroutine PHCORK

During the test phase of the Photos+ [56], a C++ implementation of the PHOTOS algorithm, it turned out that the generation becomes unstable for events produced at TeV energies. The use of PHOTOS at this energy range was then studied for the first time. The kinematics of events produced by PYTHIA 5.7 was often not precise enough, because of rounding errors for (REAL*4) floating-point variables.

The particles described by a tree-structured event record (i.e. HEPEVT common block used by PHOTOS), form decay 'branches' composed of a 'mother' – the decaying particle, and 'daughters' – its decay products. When the kinematics of a branch is not correct (the sum of 4-momentum of decay products does not match the 4-momentum of the mother particle or the particles momenta are off shell), numerical instabilities are encountered. It is not possible to perform boosts of the particles correctly if the boost parameters are large (in certain of the events where errors occurred, we found, for instance, a pion with momentum of order of TeV/c). A special routine PHCORK, which corrects these kinds of inconsistencies, have been developed and inserted to PHOTOS code[4].

The user of PHOTOS may select (at the initialisation) to activate the routine in one of the four modes:

- mode 1: no correction performed,
- mode 2: energy is corrected from mass,
- mode 3: mass is corrected from energy,
- mode 4: energy is corrected from mass for particles up to a mass of 0.4 GeV, for heavier ones: mass is corrected.

In cases, where a branch (sub-cascade starting from certain particles) has two mothers, the first mother's momentum is corrected according to the sum of its children momenta and the momentum of the second mother.

The default mode is 1, which means that no correction is applied, and full compatibility with older implementations is retained.

Correction is performed in two steps. First, all daughters' energies (or masses, depending on the requested mode of correction) are corrected to fulfil the $E^2 - p^2 = M^2$ constraint.

In the second step, the sum of all daughters' 4-momenta is calculated. The mother's momentum is corrected to this value. In cases where two mothers are present, only the first mother's 4-momentum is corrected to a value:
$P_{mother_1} = (\sum_{daughters} P_{daughter}) - P_{mother_2}$.

In 2004, for the purpose of multi-photon exponentiated mode of PHOTOS, which puts higher demands on the numerical stability of the iterated algorithm, the kinematics correction routine was extended to include a dedicated PHCORK(5) mode of operation, which is activated automatically whenever exponentiated mode of PHOTOS is chosen. It corrects momenta four-vectors of all particles taking part in photon generation: at first, the energies of all daughter particles are corrected in such a way that $E_i^2 = p_i^2 + M_i^2$; then the four-momentum of the mother-particle is recalculated to ensure energy-momentum conservation.

13.3.2 Alternative interference calculation algorithm

At an intermediate step of development, a more effective Monte Carlo weight for crude probability in interference calculation was implemented in PHOTOS 2.07:

[4]This routine became available in 1999 and can be found in the CERNLIB version of PHOTOS.

instead of doubling the crude distribution for photon emission from each charge, a flat, parallel channel was added. Although the results were satisfactory, this solution turned out to suffer from problems with stability when used with universal interference weight for more that three-body decays. We therefore reverted the code to the original implementation. For the purpose of preserving this interesting technical development and maintaining the compatibility with PHOTOS versions 2.07-2.12, released in 2003-2005, we leave this optimised code as an option to be activated by advanced users only. The two options are marked in the code by VARIANT A and VARIANT B comments, the former being currently active, the latter being commented-out. Due to elimination of numerical instabilities, we were able to lower the value of XPHCUT for variant B to 10^{-4} and perform important tests. Such tests were not possible in the default variant A, because the value of XPHCUT has to be kept at 10^{-2}.

13.3.3 Elimination of the bug in iteration algorithm

Since the version 2.0 PHOTOS used iteration of the functional photon-emission kernel to simulate double-photon emission. In theory the iteration could have been iterated more than twice, yet in PHOTOS versions up to 2.07 this would result in the PHOTOS stopping with the error message indicating numerical stability problem (negative probability). Lowering the value of the XPHCUT parameter in the alternative implementation of the crude distribution (described above) brought our attention to the fact that the reason for this "emergency stops" of PHOTOS could be due to a problem in the organisation of the iteration process, rather than numerical stability. Indeed, after (significant amount of) investigation we determined a "bug" that did not affect the results of single- and double-photon generation, yet it prevented the iteration from being executed more times. The reason was one of the variables not being cleared properly at the end of execution of the iteration. Once identified, the bug was corrected and the tests of multiple iterations have been started.

13.4 Multiple-photon (exponentiated) version

Once the problems of numerical instabilities and the bug in the iteration algorithm were resolved, we started to test the possibility of extending the PHOTOS algorithm for multiple-photon generation: three and four-photon (fixed photon number) options were added and compared [168].

In the iterative algorithm for (fixed-number) multiple-photon emission in PHOTOS, the probability of photon generation is based (at crude level) on a binomial distribution, i.e. the probability for generating and not generating a photon is calculated in each iteration. Encouraged by the precision of PHOTOS with a photon multiplicity higher than 2 (subsequent terms in the expansion improved the agreement with exact calculations), we decided to replace the binomial distribution with a Poissonian distribution for the number of photons to be generated. In this new Poissonian mode, the algorithm is no longer limited to a fixed maximal number of photons: the actual number of photons is also generated (at this "crude-generation" level). The modification set the algorithm free from the negative-probability problem as well. At a crude level, the total probability for single-photon emission in the iterated kernel does not need to be smaller than 1, because it is multiplied by the exponent of an equally large, but negative, number[5]. Because the crude distribution resembles an expansion of an exponent we call this version of multiple-photon

[5]This is trivial: if p is the probability in binomial distribution, it must be smaller than 1; however it does not need to have in Poissonian distribution given by $P_n = \frac{1}{n!}e^{-p}p^n$

algorithm an "exponentiated mode" of PHOTOS. Our approach is also close to the language of exponentiation as known in QED since Yennie-Frautchi-Suura times [169]. The exponentiated mode, implemented in PHOTOS since version 2.09, has proved to be more stable, allowing us to significantly lower the value of the infrared cut-off on the photon energy (XPHCUT) to as low as 10^{-7} (from the 10^{-2} being used for the modes with fixed number of photons).

13.5 Interference weight

In the published version of PHOTOS [144] the interference between photon emission from two sources was available only for two-body decays into particles of opposite charge and equal mass. In 2000, the interference correction was extended to work also in the case of decays into particles of distinct masses and the interference correction was re-coded to work more efficiently. For example this should prevent the program to stop in case of double bremsstrahlung and interference corrections in the decay of the very heavy $Z/\gamma^* \to e^+e^-$ state. For other channels, however, the interference was not taken into account in the calculations at all...

13.5.1 Weight correction in the decay $W \to l\nu_l$

For the leptonic decays of the W boson, PHOTOS predictions were carefully compared with the results from the first order matrix element generator [170]. It was found that even though the leading corrections were properly modelled by PHOTOS, the non-leading missing effects were relatively large. The origin of these discrepancies was understood, and the appropriate correcting weight was introduced into PHOTOS. With these changes the agreement with the matrix element calculation improved sizably [167], especially in regions of phase space populated with hard non-collinear photons.

13.5.2 Universal interference weight

In Winter 2004/2005 our attention was drawn to the fact that PHOTOS is used in BB-physics experiments, and its precision is not sufficient. At that time, PHOTOS was supposed to be used only as a "crude-level" tool for the physics processes (such as B, D, K mesons decays) for which it was not optimised: the main cause of the insufficient precision turned out to be the lack of treatment of interference for the processes being studied (see also Section 18.2.4). Motivated by needs of the B-physics experiments, a new, universal interference weight was implemented, allowing for the calculation of an interference for "any" process.

Thanks to the technical development of PHOTOS performed in 2003 to implement correction weight for the decays of W's it was rather straightforward to implement the universal interference weight for any decay channel, as expressed by formula (17) of [144]:

$$W_{multi} = \frac{\sum_\varepsilon \left| Q_1 \frac{q_1\cdot\varepsilon}{q_1\cdot k} + Q_2 \frac{q_2\cdot\varepsilon}{q_2\cdot k} + ... \right|^2}{\sum_\varepsilon Q_1^2 \left| \frac{q_1\cdot\varepsilon}{q_1\cdot k} \right|^2 + Q_2^2 \left| \frac{q_2\cdot\varepsilon}{q_2\cdot k} \right|^2 + ...}, \tag{13.1}$$

where Q_1, Q_2, ... denote the charges of the decay products, q_1, q_2, ... denote the momenta of the decay products, k denotes the energy of photon, and a summation is performed over photon polarisation states denoted by ε. Earlier, the interference weight in PHOTOS was calculated from internal angular variables and not from four-vectors; therefore such universal implementation was not trivial.

Starting from version 2.13 a new, universal algorithm calculating interference effects for all processes is present in PHOTOS. By default, as in previous versions, this algorithm (implemented internally as the Monte Carlo interference weight) is active and may be controlled using INTERF flag. To use this new interference weight, it is required that the value of the FINT parameter be adjusted in such a way that the maximum weight in the Monte Carlo rejection is increased to 2^{n-1}, where n denotes the maximal expected multiplicity of charged particles in decays processed by PHOTOS. An omission in adjusting this parameter often results in PHOTOS stopping with an error message indicating too high a value of summary weight. On the other hand, putting the value of FINT too high may significantly increase the CPU time consumed by PHOTOS because of Monte Carlo rejection not being effective. An option for having the actual value of FINT automatically calculated for each event is being considered for implementation in the future.

13.6 Other enhancements

The summary of PHOTOS parameters may be found in the preprint version of [153] (it is also present in tables **??** and **??** in the Appendix).

For some users, it is convenient to block/enable `PHOTOS` generation in some cases, for tests or because of unphysical coding of events into the `HEPEVT` common block. Flags to block/enable `PHOTOS` generation in $\pi^0 \to e^+e^-\gamma$ and in W decay into quarks were introduced. The flags are initialised in the subroutine `PHOINI`.

13.7 PHOTOS and the event record

As it is the case for many Monte Carlo event generators, PHOTOS uses event record to collect the input data, and store the results of its generation. As of now it relies on the HEPEVT even record (discussed in Section 11.1), and its FORTRAN implementation. In this section let me present the aspects, that have not yet been discussed in any of the publications, related to PHOTOS's use of the event record.

Let me start with the description of the event-record interfacing layer of PHOTOS. The fact that a multitude of non-standard, and non-compatible, versions of the HEPEVT event record is in use (they differ in the precision of the floating point variables, and the size of the arrays), we decided to introduce an "interface" layer, that is responsible for importing the data from whichever variant of the HEPEVT used by the host generator(s), performing necessary conversions, and storing the data in PHOTOS's internal event record, the structure of which follows exactly the standard HEPEVT definition (from the time of LEP2 upgrade of the standard). This internal copy of the event record, called PH_HEPEVT, may then be used by the main algorithm of PHOTOS. At the end of the execution, the data from PH_HEPEVT are copied to the HEPEVT common block, again performing the necessary conversions. The interfacing code, performing actual copies/conversions between HEPEVT and PH_HEPEVT, is separated in two, functions called PHOTOS_GET and PHOTOS_SET, therefore any adaptation of PHOTOS to a specific format of the HEPEVT common block will be well localised to the code of these two functions. Interfaces to other event records could, in theory, also be provided in that way: they would need to convert the data from the native representation no the representation based on HEPEVT standard, used in PH_HEPEVT. Let me point out that the issue of interfacing of PHOTOS with a few most popular HEPEVT formats (by means of dedicated interfacing code) was resolved by the TAUOLA-PHOTOS-F environment, discussed in Chapter 16.

Having discussed the interfacing of PHOTOS to an external event record (whichever

format of HEPEVT was it), let me now proceed with description of PHOTOS operations performed on its internal, PH_HEPEVT event record common block (executed in the call to PHOTOS_MAKE). The core algorithm of photon generation, as implemented in PHOTOS and discussed in the papers, generates bremsstrahlung for a single, elementary particle decay process. For the cascade decays of particles, the photon generation algorithm should (typically[6]) be applied to all particle decay processes that involve charged particles. For that purpose an algorithm was implemented in PHOTOS to search through the whole decay cascade (starting from a point indicated as a parameter to the call of the PHOTOS routine) and identify all "branching points" - potential points where the photon-emission algorithm might be applied. Integrated into the search procedure are the procedures that verify the grammar of the event record: the searching algorithm assumes the standard use of the mother-daughter pointers in the event record and the tree structure of the event record from the point where it starts the search, hence it is vulnerable to inconsistencies or loops. The verification procedures prevents PHOTOS from generating photons in the parts of the event record where the meaning of the entries is unsure, namely for the entries with the status code indicating the event-history fragments of the event record. By employing these routines PHOTOS is a powerful tool adining to detect problems with the structure of the HEPEVT event record, and indeed it was used as such in many experimental simulation chains.

For each of the branching points (particles that decay with at least one charged product), the core photon-generation is the applied in a loop. For that purpose, the fragment of the event record starting at the branching point is copied to a PHOEVT data structure, which, again, has the format that follows the concepts of the HEPEVT event record. The branch, which in the case of the topmost particles in the cascade may contain a lot of sub-branches, hence may include many particles, need to be boosted to the reference frame of the decaying particle. Often, its kinematics need to be checked and corrected, beforehand (using the routine presented in Section 13.3.1) to eliminate numerical problems. Then, the core photon-generation algorithm may decide to add one (or more - if multiple-photon option is activated) photon to the decay (the actual algorithm used to generate photon(s) is described in, for instance, [5]). If any photon(s) were added, the kinematics of all particles in the "branch" need to be recalculated to take into account the impact of the photons. Adding the photons to PHOEVT may also be a non-trivial task: the encoding of the tree into mother-daughter pointers requires that the photons are placed in the list consecutively to the particles being direct children of the top particle in the branch. In the case of a branch containing a cascade of particles, the place for the photon(s) need to be prepared in the list at first: all the entries below need to be shifted, and their pointers adjusted appropriately. Then, the modified branch (in PHOEVT) needs to be put in the place of the original branch, in PH_HEPEVT. This requires that all particles in the branch are boosted back from the reference frame of the decaying particle to the laboratory frame. And, again, the place for added photons need to be made in PH_HEPEVT, which involves shifting of all entries laying below, and adjusting all mother-daughter pointers (in the whole event record).

Lack of the dynamic data structures, and pointers in the FORTRAN codes, and the assumed structure of PH_HEPEVT make the event-record related parts of the PHOTOS code particularly complicated and unclear. Lack of the C++ event record standard and the need to fall-back to the index-based HEPEVT made these

[6]Yet only for the processes for which radiative corrections have not yet been included. While the question is not relevant for `HERWIG`, since no photon bremsstrahlung is implemented, it can become an issue in `PYTHIA`, for some specific processes. Indeed, one can generally switch off the photon radiation off leptons (by setting a high limit on `PARJ(90)`, cf. [34]); however, one cannot do it on the basis of specific processes and/or decays.

steps uneasy also in the C++ implementation of PHOTOS [56]. Having proved the flexibility of the HEPEventLib approach (Section 11.6.9), we could now re-consider a new C++ implementation of the PHOTOS algorithm using more suitable internal representation for the event record data (corresponding to PHOEVT and PH_HEPEVT) and delegate the task of translation between this internal representation and event record standards (HEPEVT) to the HEPEventLib routines. We believe that the PHOTOS algorithm might this way be expressed and implemented in much clearer, modular and flexible way, allowing for its further evolution, and resolving, once for good, the problems with interfacing of PHOTOS to generators such as HERWIG or PYTHIA, which use a variety of non-standard grammar for the HEPEVT event record. Again, we believe that the use of solution based on the concepts of HEPEventLib will allow to separate the problems related to the event-record compatibility from the code of PHOTOS.

A short discussion of PHOTOS in context of C++ event record (being the input/output) deserves to be put here. In the layer-based approach, as postulated in Chapter 10, each generator contains the trace of its activity (event history) in one (or many) layer(s). So far, PHOTOS does not introduce a notion of "layers", but rather modifies the event record directly. The actual modifications of the event record introduced by PHOTOS may involve significant number of objects: not only photons are added (in the whole decay cascade), but the kinematics of the decay products in sub-cascades are changed. The kinematics of particles is also altered by the kinematics-correction routines. One could consider an approach, in which the event record is used at all the stages of PHOTOS execution: the PHOEVT branches in processing of decays of subsequent particles could be represented as individual layers in the event record, and all relations could be maintained. In such approach, the availability of the "complete trace" would certainly be interesting and convenient for the authors of PHOTOS for the purposes of debugging and study of the algorithm itself (such as the studies of traces of the NNLO effects in PHOTOS). For a typical event for the LHC studies, the number of layers and instances of objects introduced by PHOTOS would be huge: the kinematics of certain particles could have been altered many times, by the generation at subsequent steps of the decay cascade. In the practical applications, a "collapsed view" would be much more pragmatic: PHOTOS might use a single layer, which would contain the instances of all modified (and added) objects after the full generation step. In such approach, the state of the event "before" and "after" the execution of PHOTOS could still be resolved, while keeping the size of the event record at a reasonable size.

Ultimately, let us consider a possible scenario for re-implementation of PHOTOS with the HepMC event record, as already postulated e.g. in [109]. Even though the use of dynamic data structure would simplify the activities related to adding the photons to the event record, the verification and sanity-checking algorithms, excluding the loops, would probably need to be kept in PHOTOS, to decode the status information, etc. Lack of the layered structure imposes the constraint on the form in which the output data from PHOTOS would be generated: the existing particles will rather be replaced by their updated instances, as it is the case for HEPEVT right now. A significant concern is stability and wider adoption of HepMC in the HEP, theoretical and experimental, community. Therefore a more generic approach, such as the one postulated above (based on abstract event record abstract layer such as HEPEventLib) would probably be more applicable in the current situation.

Chapter 14

TAUOLA and the universal interface

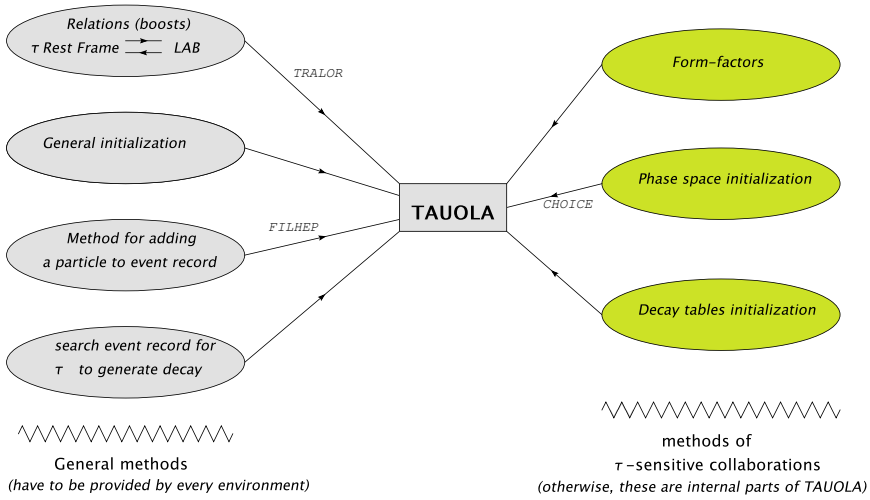
The TAUOLA [27, 29, 28] Monte Carlo generator package is a library of the decays for the τ-lepton channels, widely used by HEP experimental collaborations. In this chapter I am not going to repeat the phenomenological and theoretical description of the τ-lepton decay process, and its modeling in TAUOLA - this topic was discussed in length in the PhD thesis of T. Pierzchala [3]. Nor will I repeat the details of the application of TAUOLA in scope of the Higgs boson studies and modelling of the spin-correlations mechanism, as these were discussed in the PhD thesis of M. Worek [2]. To some degree I participated in these activities, mainly at the stage of construction of the tools that were used at the later stages of analyses. Instead, I am going to describe the issues related to the application side of TAUOLA, and interfacing it with event records, namely the TAUOLA Universal Interface. I will also discuss the reasons that led to the creation of the TAUOLA-PHOTOS-F package, discussed in chapter 16, which is the need to maintain versions of the code that contains various physics initialisations originating from experimental collaborations.

14.1 TAUOLA as a component

The modular organisation of TAUOLA's code make it easy to extend and adapt to the need of the experimental collaboration. It is however striking that TAUOLA fulfils the requirements of component based programming (see also Sect. 4.1)! The data exchange is supposed to be organised in interfaces, with well-defined specification (discussed in section 14.2), that need to be implemented to integrate TAUOLA within an architecture. The "native" architecture of TAUOLA is the one of the KORALB [171] and KKMC [55] generators. A general-purpose interface (to the HEPEVT event record) is provided by the TAUOLA Universal Interface, described in Section 14.3.

Let us now point out a peculiarity of the KKMC-TAUOLA interface, which is dictated by the requirements of the treatment of the full spin correlations in τ lepton decays, the topic discussed further in the next section . Unlike the case of typical approach to event generation, the event generated by KKMC with TAUOLA is not constructed sequentially, in separated computational steps. KKMC needs to generate partial information about the production of the τ leptons at first, then call TAUOLA to generate their decays, get the decay information back from TAUOLA, and only then finalise the construction of complete event (calculate the spin weight and total Monte Carlo weight of the event and accept or reject the event). Such organisation of processing is, in general, not possible in the most of event simulations

Figure 14.1: Methods required by TAUOLA.



based on the architectures with central event record, and independent generation steps.

Recently, in the course of development of the TAUOLA interface to the ATHENA framework [72] we encountered a particular type of problems related to organisation of the complete simulation. Because of certain design decisions concerning the architecture, and the need to pass the data through the HepMC event record, a new requirement for TAUOLA was expressed. Up till now, TAUOLA was usually used together with generators such as JETSET, which were responsible for generating decays of resonances, such as η, produced by TAUOLA as decay products of the τ lepton(s). In the architecture of ATHENA, however, to generate the complete decay of τ lepton(s) would require multiple translation of event record data from HepMC (native event record) to HEPEVT (for TAUOLA and its universal interface), then again to HepMC, in order to record the primary decay products of τ's, then to LUJETS/PYJETS to perform decays of particles generated by TAUOLA, such as η's, then again to HepMC. In order to simplify this, TAUOLA and its Universal Interface needed to be extended to perform the decay of resonances, thus to eliminate cumbersome translations of event record data.

14.2 TAUOLA application and context of use

TAUOLA is a library of the lepton decays, and as such it is not used as a standalone Monte Carlo event generator. Instead, it should be made a part of a large generator system, integrating (or interfacing) it with the τ production such as KKMC [55]. The actual way of interfacing TAUOLA with other generators is, however, dictated by physics, namely by the way in which spin correlations are treated. To make use of TAUOLA, the user needs to provide methods (subroutines) that implement the handling of the information between the "host" event generator (or the main simulation) and TAUOLA. The application model is presented in Figure 14.1.

In every use case, the following general routines (grouped on the left side of the Figure 14.1) need to be implemented:

- A general initialisation method which sets up the main physical and technical parameters of TAUOLA

- TRALO4: a subroutine that define the relation between the rest frame of the τ lepton and the laboratory rest frame.

- A method that finds the τ leptons in the event record and hands them over to TAUOLA. In order to treat the spin correlations appropriately, the pairs of particles: $((\tau^+, \tau^-)$ or (τ, ν_τ) , as produced by the host generator, need to be given to TAUOLA. This task may be delegated to the TAUOLA Universal Interface (described below), which searches the τ pairs in the HEPEVT event record; the interface provides the input for the abstract TRALOR method as well.

- FILHEP: a method that appends the decay products generated by TAUOLA to the event record.

These methods needs to be provided either by the host generator itself (as it is the case for KKMC, KORALZ, KORALW, KORALB, HERWIG) or by a dedicated interface. In the case of the TAUOLA Universal Interface, a special way for providing event data to the TRALOR subroutine is provided.

For experimental collaborations that perform τ-sensitive measurements, additional methods (presented on the right side of the Figure 14.1) may be of particular interest.

- the parameterisation of the form-factors: these are usually parts of collaboration fitting packages, and their callibration involves the experimental data (such as the data from low-energy e^+e^- scattering [3]).

- CHOICE: the subroutine responsible for the initialisation of the phase-space Monte Carlo generator (in particular for the presampler and the treatment of resonances)

- Particle Decay Table initialisation: the tables which define, among others, the flavours of particles for each decay channels; the actual list of decay channels according to which the τ decays are generated may also be defined there. For the experiments such as Belle, or BaBar, custom (rare) decay channels are added here.

For a typical LHC setup[1] there is no need to re-define these methods of TAUOLA - the default implementations should be sufficient.

To generate the decay of (a pair of) τ leptons TAUOLA's subroutine belonging to the DEKAY or DEXAY family needs to be called. The choice of the family of routines depends on the physical content of the generator and the mechanism for treatment of the spin corellations. The diagrams in Figure 14.2 presents the way these functions need to be applied.

For the full treatment of spin correlation, the interface based on the DEKAY routine needs to be used. This mechanism is also refereed to in the TAUOLA documentation and code as KORALB type. For the full treatment of spin correlations being possible, one cannot separate the production and decay processes, i.e. special organisation of the interface is needed. At first, the host program (such as KKMC or KORALB) generates an unpolarised pair of $\tau^{\pm}$. Then, TAUOLA's DEKAY routine is called twice to generate the decays of the subsequent τ's; as a result of this call not only the decay products, but also the polarimetric vectors h_1^μ,h_2^μ are generated. The polarimetric vectors are then returned to the host program, where they are used in the calculation of the spin weight of an event (i.e. after the generation

[1] probably with an exception of the LHCb, which at some stage, will probably also make use of these extended functionality, once the data-taking will start.

of the τ decays the control needs to be returned to the program which generated the process of the τ lepton production. Only after the final weight of the event (which takes into account the information generated in DEKAY, yet calculated by the production generator) is known, the event may be accepted or rejected. For the accepted event, the decay products are boosted to the laboratory system and stored in the event record.

Due to the physics-motivated needs for the organisation of the combined τ lepton production/decay generation as outlined above, the DEKAY-based interface is often hard to combine with general-purpose host generators (such as PYTHIA or HERWIG), which either does not consider the spin degrees of freedom in the calculations, or apply other (approximated) approaches, not based on the technique of polarimetric vectors. For these generators, which separate the process of production and the decay of the τ lepton, a group of the DEXAY functions (also refereed to as KORALZ-style) was created. In the DEXAY-based interface, the production ("host") generator is responsible for providing the information about the polarisation of the τ lepton pair - the polarisation vector are the parameters passed to the DEXAY routine. In this case, TAUOLA generates the decay of 100% polarised τ's. Despite the transverse spin correlations being neglected, this approximated approach gives good results for ultrarelativistc τ's.

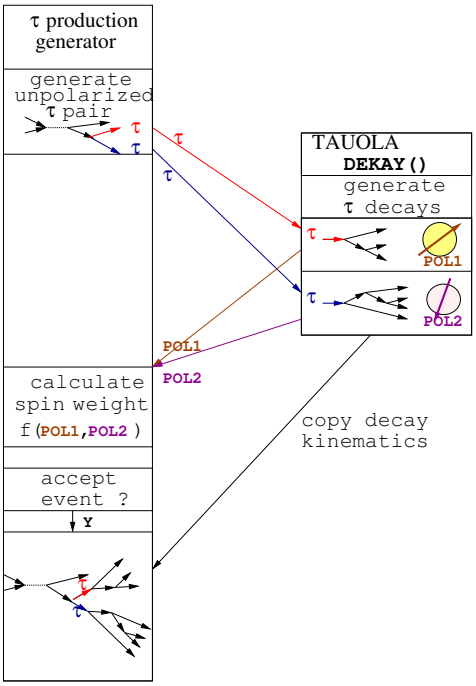
14.3 TAUOLA universal interface

Despite the fact that TAUOLA is being widely in many experimental collaboration's software packages, such as QQ[139], its application to the new studies and software environments requires a standard interface to one of standard event records. HEPEVT being the most feasible candidate. The TAUOLA Universal Interface was developed for that purpose and published as a part of the TAUOLA-PHOTOS-F package, described in more detail in chapter 16.

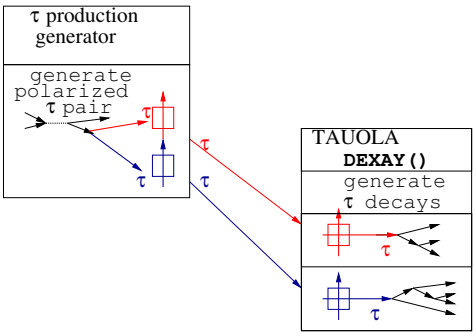
The universal interface application scheme was limited to the option with DEXAY() routine, which means that the full spin correlations could not be taken into account and only the approximate treatment would be possible, at least as far as TAUOLA is concerned[2]. Moreover, the majority of the general-purpose generators does not deliver the spin-related information in the applicable form (i.e. polarisation vectors), or does not use spin information at all. For such cases it is, however, sometimes possible to calculate the spin information from the the HEPEVT event record, and make use of it in TAUOLA. A special method that extracts the spin information from the (HEPEVT) event record and transfers it to TAUOLA was implemented in the Universal Interface. The theoretical mechanisms allowing to include spin effects in e.g. the $Z/\gamma^* \to \tau^+\tau^-$ process are covered in [96, 172]. It is also one of the main topics of the PhD thesis [2]. The application of these mechanisms to the TAUOLA Universal Interfaces are also covered in [109], therefore it will not be repeated here. In fact, the TAUOLA Universal Interface may be considered as a separate software project, to some degree independent from both the specific problem of τ production and its decay. The aim of the interface is not to replace the matrix element calculations, but rather to provide a method of calculating/estimating spin effects in the cases where these would not be taken into account at all.

[2]In the TAUOLA Universal interface there exists also an extended version of the standard interface that includes the complete spin effects for τ leptons originating from the spin zero particle, i.e. in $H/A \to \tau^+\tau^-$ decay [172, 173, 174].

Figure 14.2: DEKAY and DEXAY interfaces of TAUOLA



(a) DEKAY()



(b) DEXAY()

14.3.1 Event-record related issues

The issue of encoding the tau decay products into the a (tree-structured) event record is not as complex as in the case of PHOTOS (see Section 13.7): the status code of the τ's are changed, the decay products are appended at the end of the event record, and the relationship pointers are updated. The decays of the τ's are generated in the rest frame of the τ's, therefore the four-momenta of the decay products need to be transformed between the τ's reference frame and the laboratory reference frame before and after the actual decay generation step. In a general (i.e. non-approximated) these Lorentz transformations need to be treated with care in order to preserve the information about the spin quantisation axes (the problem manifests itself, for instance, for $e^+e^- \to Z/\gamma^* \to \tau^+\tau^-$ process, and the proposed solution is described in [101, 175]). The task of performing the Lorentz transformation of any four-vector between the rest-frame of specified τ and the laboratory frame was therefore dedicated to the TRALO4 subroutine, that needs to be provided as a part of the interface to TAUOLA. TAUOLA uses this routine not only to boost the products of the τ's decays, but also to transform the spin quantisation frames (used in the interface to KKMC generator). Due to such organisation, TAUOLA does not require that any event record (of any defined structure) is available. The only information that needs to be transferred from the τ-producing generator is the relation between τ's rest frames and the laboratory frame, provided by means of TRALO4. No assumption concerning the structure of the event record needs to be made, no data describing the momenta of particles need to be transferred! After the generation of the τ's decay is completed, the decay products are stored in TAUOLA's internal data structure and it is in general, again, the role of the interface (or the host generator) to transfer them to its own event record data structure (or make use of them in other way). TAUOLA itself provides, however, a routine that stores the decay products in the HEPEVT event record: calling DEKAY/DEXAY functions with the first parameters equal to, respectively, 11 or 12 performs this operation for the first and the second τ.

The organisation of the TAUOLA input and output by means of external interfaces may seem to have eliminated the problems of the event record interfacing of TAUOLA, which proved to be complicated for the case of PHOTOS described in Section 13.7, where these issues needed to be resolved explicitly in the PHOTOS code. In the case of TAUOLA, this task was delegated to the interface layer, namely to the TAUOLA Universal Interface, which takes over the task of resolving the complexity of HEPEVT interfacing.

The organisation of the HEPEVT data exchange in the interface met, again, the same class of problems as for the case of PHOTOS. Even though appending the generated particles was not a problem (the products of τ's decays may simply be appended at the end of the HEPEVT event record list), the algorithm that locates the correlated pairs of τ leptons in the event record needed to employ a variety of grammar verification procedures, to enable the interface working with generators working with non-standard variants of the HEPEVT common block. During the MC4LHC Workshop[176] these issues were discussed and it was found, thanks to interaction with the users, that the TAUOLA Universal Interface works with all 3 basic options for HEPEVT filled by PYTHIA (`with MSTP(128)=0,1,2`)[34], as well as with HERWIG [177]. In the distribution of the Universal Interface we keep only the working interface with PYTHIA, the working example with other installations can be found e.g. in [133].

The question of interfacing TAUOLA to the C++ event records such as HepMC remains open. Such interface to the existing FORTRAN version of TAUOLA may well be written, taking the Universal Interface as a starting point. On the other hand, to assure the future development, the code of TAUOLA is likely to be rewrit-

ten in C++, if only experimental collaboration expresses the need for such translation - all tools required to perform such translation are already in place. The use of component-based approach is a likely solution, hence the task of interfacing to event records will again be delegated to external interfacing modules.

Let us stop the discussion here and, instead, let us refer the interested reader to [178, 97] and [133].

14.4 TAUOLA in experimental collaborations

An essential part of TAUOLA that needs to be controlled by experimental collaborations is the physical initialisation of the code. In particular, when used as a part of large collaboration software systems, which combine complex data-fitting packages with Monte Carlo simulations, the initialisations often needs to be tuned to the actual needs: branching fractions of decay channels, masses and widths of resonances, callibration (and sometimes also parameterisation) of hadronic currents are the subject of changes. This way, the code may evolve out of the control of the authors, enriching its physical content and value - new parameterisations or callibrations, related directly to experimental data, are incorporated in the version of code maintained internally by the collaboration. These enriched versions of TAUOLA often migrate to other experimental collaborations - usually as parts of larger packages, such as QQ [139], to enable comparisons and combining of experimental results between experimental collaborations.

Throughout the decade of TAUOLA's use, three major variants of its code were identified and archived by the authors: the original author's version of the code, as published in CPC [171], the version originating from the CLEO collaboration [179], and the version originating from the ALEPH collaboration [180]. The intention of the TAUOLA's authors was therefore to preserve these alternative versions of physical content, while making it possible to be used in other projects, by means of the interfaces not implemented in the original collaborations' code. This need motivated the creation of the TAUOLA-PHOTOS-F package, described in Chapter 16.

Part V

Methods and Tools

Chapter 15

Programmer's tools

Growing complexity of the simulations requires the use of more and more advanced tools aiding the process of design, implementation, validation, testing and maintenance of their computer codes. In this chapter I gather the issues related to the organization of the simulation codes into libraries and summarise my experience with code building, and software management tools.

The topics signalled in this chapter would deserve a separate, dedicated monograph. To my knowledge, such description (focused in particular on the requirements of scientific applications, with applications in the Linux operating system) has not yet been published; typically, the experience need to be built by miss-or-hit method, and assembly of bits of information from many sources. For a physicist non interested in technicalities of operating system and compiler design, acquiring the appropriate knowledge may indeed be tiresome. Such documentation is being completed and is planned to be published [181].

15.1 Libraries and header files

Contrary to the practises of many authors of the FORTRAN77 codes, who often distribute their projects in form of solid units contained in a single source file (sometimes counting tens of thousands of lines), it is much more pragmatic to separate the code into smaller files, containing functionally and logically grouped functionality. The requirement of the C/C++ to present the declaration of the function (type, variable, class ...) before its use in a file, naturally led to a practise of separating the declarations and definitions. The use of a single-source-file based approach is not acceptable anymore. Instead, following the mechanisms used in majority of operating system nowadays, the binary, compiled version of the code of one or many modules should be stored in libraries, accessible to other projects. Not only the re-use of the code is made easier this way - the widespread use of dynamic/shared libraries mechanisms allows to dramatically reduce the size of binary executable programs stored on the disk, but also in memory - during the execution. In the case of C/C++ modules, delivered in the form of a (binary) library accompanied by "header" files containing function definitions, it is possible to provide a "closed-source" modules, which are possible to be re-used by other programmers, yet their source codes needn't be disclosed. Such practise, contrary in it's idea to the openness of scientific computer codes allows, for instance, to obtain highly-optimised numerical computation libraries from hardware vendors who will to protect their intellectual property. On the Linux platform, used mostly in the scientific environment, the closed-source libraries are almost non-existing. Nevertheless, the solution based on (mainly dynamic/shared) libraries and header files only being installed

is used for the vast majority of the system components and programs (including standard C/C++ libraries, mathematics, memory management, graphics, etc).

Contrary to programming languages such as Java, Pascal, or Perl, a binary C/C++ library does not include the information about the "interface", i.e. the list of defined symbols, functions, classes, etc. Instead, the (public part of the) interface is specified in a separate "header" file, which contains the definitions of the symbols that are supposed to be used. The interface specification is often refereed to as Application Programmer Interface, or API. It is possible to have a library accompanied by more than one sets of header files, e.g. the one with specification of all symbols (as required to compile the library), which is also refereed to as "private interface", and one with selected symbols only ("public interface", API). Header files are therefore a natural, self-documenting and directly usable specifications of the interfaces.

Since decades the libraries related to HEP computations are made available and maintained i.e. at CERN; the most prominent of them is probably CERNLIB, the full documentation of which may be found in [26]. This library (with a set of accompanying auxiliary libraries) provided the HEP community with routines callable from the FORTRAN codes, that performed most commonly used tasks (e.g. phase space Monte Carlo generator, integrators, minimisation packages, numerical calculation functions) and provided mechanisms necessary for HEP data processing (histogramming, memory and file management). A large part of CERNLIB was the set of universal Monte Carlo event generators, the code of which was maintained (in contact with the authors) by the CERNLIB librarians. The role of CERNLIB is marginalised now due to various reasons: the LHC experimental collaborations maintain their own version of the generator codes - part of the effort of centrally generators library is fulfilled by the LCG's GENSER project [66]; the activities related to statistical analysis, numerical calculations, histogramming, etc, performed earlier by packages such as NAG, HBOOK, ZEBRA or PAW, are now done using the ROOT package [60]; detector simulations are now based on the GEANT 4 package [43], managed independently by its developers.

The concept of function libraries may be conceptually extended to class libraries and frameworks used widely in C++. From technical point of view, they may either be contained in standard, binary libraries (as it is the case for ROOT [60], Ptolemy[74], or Qt[182]) or have other forms, such as template libraries (STL [183]). Resolving the issues of backward compatibility (at the source and binary level) and cross-dependencies between new versions of the libraries is amongst the most time-consuming activities for the authors of the packages, but also for the librarians.

15.2 Build systems

Increasing number of source files, header files and dependencies between them, and use of multitudes of shared and static libraries, various versions of which may (or may not) be installed in a variety of paths in the filesystem, specification of options to compiler and linker tools, on a variety of flavours of the operating system makes the organisation of the build system for a large software project a very challenging task nowadays. Even though the standard tool, *make[184]*, is used commonly to organise the build system for the projects on Linux and UNIX systems, it hardly specifies the way to express the most basic dependencies and rules governing the use of compilers, linkers, preprocessors or other tools (the names and options of each need to be specified explicitly).

In the projects such as KKMC[55] or TAUOLA[27], or KORALZ[165], the use of *make* command was reaching much further than building the executable programs from source codes and libraries - the command (and the build system) was also used

to run the simulation (also in batch mode, on the clusters of machines) and collect the results of the simulations. Similar philosophy is also used in MC-TESTER, described in Chapter 17, which has its generation and analysis infrastructure organised this way.

The *make*-based build systems, crafted "by hand" are often not sufficiently flexible when a project is composed of many sub-directories, and it needs to be restructured. For instance, changing the options passed to the compiler may require editing the *Makefile*'s in all the subdirectories of the project. For KKMC [55] a template-based solution was employed, with all *Makefile*'s being re-created in the project upon execution of a dedicated program. The concept of such higher-level tool that produces working *Makefile*'s based on a higher-level project description data is not genuine - it is used widely in the software projects; some of the available standard approaches will be described below.

During the work on the TAUOLA-PHOTOS-F environment (described further in Chapter 16), the build system of the TAUOLA and PHOTOS projects was reorganised to enable central control over the settings (and the names) of the build tools (compilers, preprocessors, linkers), and simplify the contents of the *Makefile*'s. The solution I worked out also allowed for easier portability of the project: all platform-dependent settings were gathered in a single *make.inc* file, included by all other *Makefile*'s in the project, and multiple versions of that file were made available for various platforms (operating system/compiler versions). The same concept of central management of the settings in the *make.inc* file was later used in the MC-TESTER project.

The solution used in TAUOLA-PHOTOS-F and MC-TESTER were, to some degree, inspired by the solution used in the GNU Automake and GNU Autoconf tools [185, 186]. These packages, used already for long time in large Open Source projects, provide a solution to the majority of the problems raised at the beginning of this section: portability to other platforms, automatic detection (and option for specification) of installed tools, and their appropriate options, detection of available libraries, verification of their functionality and versions, and generation of (complex!) Makefiles from a high-level "template" files. At the time when the TAUOLA-PHOTOS-F package was developed, the complexity of the *autoconf* tool and lack of available introductory documentation discouraged me from employing it as a build system for the project. Moreover, it was essential that the *Makefile*'s distributed with TAUOLA-PHOTOS-F were ready-to-use (i.e. the availability of autoconf/automake should not be required, even though they are standard elements of the Linux installation) and simple to edit. The *automake* and *autoconf* tools were, however, later evaluated in context of tests of the KDevelop IDE, described below. In the pilot project, we succeeded in transforming the mFoam generator project [187] to the build system based on *automake/autoconf* - automatic detection of CERNLIB and ROOT installation in the autoconf were implemented as a proof of concept.

Another popular tool for management of the build system is qmake, which is an integral part of the Qt framework[58]. The unquestionable virtues of *qmake* are its support for portability, which allows to organise build systems even on the Windows platform (i.e. using *nmake* as building backend), and extreme simplicity of the syntax. The most severer drawback is, however, lack of possibility of integrating custom makefile targets, hence excluding the possibility of advanced used to, for instance, organise the batch-system running environment, etc.

CMT [188] is the tool used for management of versions and building the packages in the ATLAS collaboration. Being developed in the HEP community, it is less known elsewhere, which is its main deficiency. It is focused on the concept of packages and management of their versions and releases. Other tool, SRT [189], developed by BaBar collaboration is also sporadically used in some projects (it was

used in ATLAS before).

15.3 Project management and development tools

The development and management of the software project, even those of not so large scale, such as some event generators, may be aided with the use of standard tools available on Linux/UNIX platforms. I found the following group of tools particularly useful and interesting in context of various project development

Code version management

The most widespread tool in this domain, available also on the Windows platform, is *CVS* [190]. It allows to record and track the changes made in each file of the project, and maintain versions, releases and branches of the code. It also provides functionality needed for maintaining parallel development of the same project by a group of authors (i.e. file and repository locking, resolving conflicts). CVS is based on the concept of a single, central repository for the code, from which the developers may "check out" requested version of the project to their working directories, then "check in" their contribution to the repository, so that the other developers may gain access to it. The feature I found particularly useful is the possibility of setting up the repositories in a very easy way (on a local disk, e.g. in the home directory), then accessing it smoothly from any remote machine over encoded, reliable ssh connection. This way the creation of "ad hoc" repositories available remotely for groups of users (hence enabling collaboration on the project development) is made effortless and does not require the intervention of, say, system administrator. On the other hand, dedicated CVS servers are managed at CERN and available for projects developed by CERN users. The users of these servers profit from high-reliability and safety of their code - the data is regularly backed up, and web-based frontends for browsing the code in the repositories are available. The use of CVS is not limited to storing computer codes - I find it also very useful to maintain any results of work that requires versioning, in particular scientific publications.

Recent years brought alternative code management systems - the most serious rival of CVS being *Subversion* [191]. Subversion, described by some as "better CVS", is supposed to take over the user base of CVS by exploiting all of its advantages, while resolving many problems and limitations of CVS, the most important of which is maintaining the versions of directories, symbolic links, copies, renames and adding arbitrary metadata. Growing popularity and user base in the Open Source projects makes Subversion a serious choice for code version control system.

Centralised code version management systems such as CVS or Subversion are not well suited for the projects with large number of developers, such as the Linux kernel. For such projects distributed version control systems such as BitKeeper, arch or git are used [1].

15.4 Debuggers and profilers

Debuggers and profilers are native part of software development toolbox on Linux/UNIX systems, yet their does not seem to be very popular amongst the programmer. Simplistic, hard to use, command-line interfaces of gdb and gprof, does not encourage the use of this tools. The availability of the graphical interfaces, such as *ddd*, that allow for easier code inspection and easier interaction with the tool during

[1] For more details about these systems you may refer to the appropriate articles in Wikipedia[10].

the debugging session already improved this situation. Still, however, the classical debuggers does not bring much help in locating the source of the bugs such as memory leaks, use of initialised data, dereferencing of pointers that are not valid anymore, writing past the end of the allocated data segment (buffer overruns) or race conditions in multi-threaded applications. The symptoms of these errors may not indicate the actual source of the problems. They may however be easily pinpointed using the *Valgrind* tool [192]. Valgrind, which I consider as one of the most valuable programmer's aiding tool, is based on the concept of a virtual processor, being simulated at the hearth of the tool, used to execute the (binary) code of the program. The tool may intercept and localise all of the above mentioned types of errors, indicating the nature of an error, precise location of related source code lines, number of occurrences and gravity. As the debugged program is run on the simulated processor, its performance is usually degraded by an order of magnitude, yet the use of optimisation mechanisms (it is possible to specify the portions of code that is known to perform well, and may be executed on a real processor) allows for sensible debugging of even large, interactive applications. Valgrind was an invaluable tool during the development of at2sim, where elimination of (even minimal) memory leaks was essential to make the modelling of larger systems possible at all.

In the large computational projects it is essential to make an effective use of the computer resources. Profiler tools allow to identify the critical fragments of the code where significant amounts of CPU time is spent, so that these fragments may be optimized. Event though classical profiling tools, such as gprof, may be employed for such tasks (they in fact rarely are - the developers often limit themselves to activating maximum optimisation level of compiler only), the use of new profiling tools such as KCacheGrind (graphical interface to a part of Valgrind) and OProfile[83, 82] may be more effective. The possibility of tracing all the activities of the operating system using OProfile was the one I considered to be extremally useful for studies of the performance of the ATLAS Data Flow System - it clearly demonstrated where (and how) the CPU resources are used; it also allowed me to explore and understand all the mechanisms involved, ranging from the ones at the lowest level of the operating systems, up to the high-level multi-threaded applications of the ATLAS Data Flow Software.

15.5 Documentation tools

Maintaining up-to-date version of the documentation of the library is critical for projects that evolve; it is also one of the most boring tasks. The existence of the automatic tools capable of generating reference documentation based on the source codes is appreciated by all developers nowadays. Commenting the code (following some conventions) is sufficient to provide a documentation of functions' parameters and description of their application, and may be extracted automatically by the tools. One of the most flexible and widely used tools of this type is Doxygen[193]. Similar functionality is also provided internally by ROOT [60] for all classes developed in its framework. An important issue I wanted to stress here is that the availability of sole automatically-generated code-reference documentation (in particular generated from the code not having sensible descriptions of functions and parameters) **MAY NOT** be considered as sufficient documentation of the package. Huge class reference documentation, such as for instance [129], with no introductory documentation, overview and classification of information only lead to confusion and discourage the use of a package! The use of automatic documentation generators should therefore be treated as complementary to providing solid documentation of the packages; including proper (and maintaining up-to-date) meta-information in the code comments is essential as well. My experience with Doxygen documentation,

gathered in scope of the JCOP Framework Project [194] indicates that maintaining such documentation also requires significant effort, if the project evolves dynamically.

15.6 Diagramming tools

In the classical Software Development Process to assure the quality of the project much attention is put to the design of the project organisation in terms of classes and relations. At this stage UML modelling and ER (Entity-Relationships) modelling tools are often exploited to draw the prototypes of class hierarchies. Even though such approach may not always be applicable to scientific software projects[2], the techniques may be used as a complementary documentation of the projects. For the case of at2sim we evaluated one of commercial UML modelling tools; we concluded that the effort put in the creation of the UML diagrams at the stage of dynamic changes in the project is not justified: the created models were becoming out of date very quick (see the footnote!) due to experiments and redesigns that were required. The fact that the costs of the tool were significant, and its performance (on the machines available to us) was poor, we abandoned the use of the tool at that stage of at2sim development. Later, I evaluated two Open Source tools: Umbrello UML Modeller [195] from the KDE project, and *dia* [196] diagram drawing package. An example of a diagram generated by Umbrello is presented in Figure 11.6.

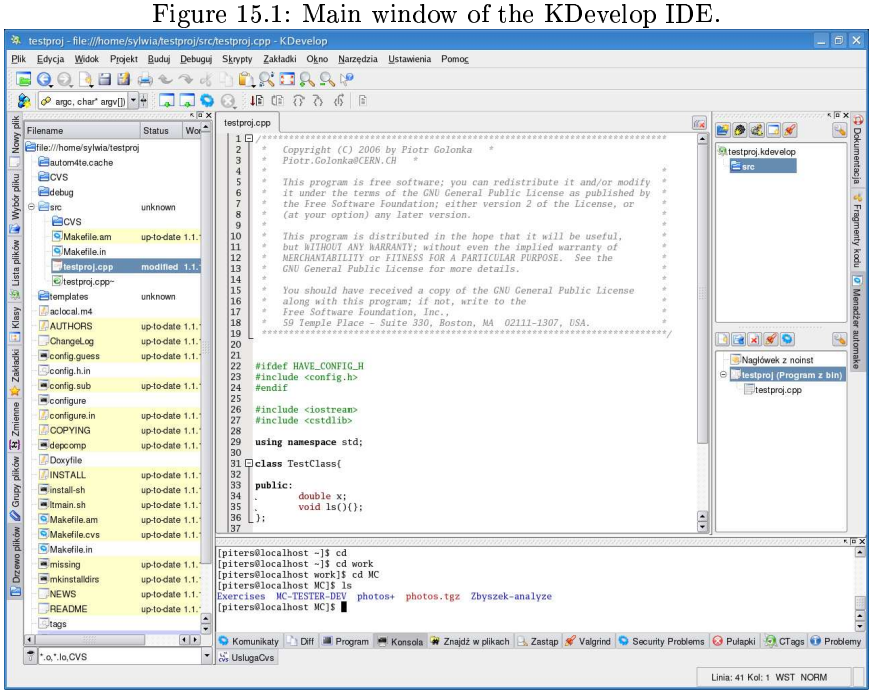
15.7 Integrated Developer Environments

Software development on the Linux/UNIX platform was traditionally performed using text-based tools, rather than Integrated Developer Environments (IDEs), which became so popular on the Windows platform in particular in context of Rapid Application Development (RAD) projects. The only integrated tool used by many developers was Emacs; even though it could be adopted to many tasks related to software development (such as program debugging, access to CVS), its specific conventions of use made it not as simple and comfortable in use ad Windows-based tools. Recently, however, a number of IDE's for C/C++/Java development on Linux/UNIX platform became available - both commercial and Open Source; amongst the most popular of them are Eclipse, KDevelop, Java Studio Developer, Kylix (Delphi) or Anjuta. Out of these products I gained the most experience from the use of KDevelop : an IDE from the KDE project. A snapshot of the main window of KDevelop is presented in Figure 15.1. The authors of KDevelop succeeded in integration of a vast set of software development tools; among others, graphical frontends are provided to various build management systems (automake/autoconf, qmake), code versioning (CVS, Subversion, ...), debugging (gdb, valgrind), advanced documentation browser (with cross-referencing) and generation tool, editor with syntax-highlighting code-completion and hinting.

We evaluated KDevelop in context of development of the mFoam project [187]. We had found that, at that time, the version of KDevelop was already offering an interesting set of features, yet its stability, usability and documentation were still not sufficient to recommend it for use in the projects. Since that time, KDevelop evolved significantly and the majority of the issues identified by us during its evaluation were already fixed. A comprehensive evaluation of a recent version of this IDE would need to be repeated now to draw sensible conclusions.

[2] As discussed in part *I*, for certain scientific projects the aim is to experiment and explore the possibilities, rather than to provide a stable consumer-ready product; gathering the requirements and designing the definitive architecture of such projects are by definition impossible.

Figure 15.1: Main window of the KDevelop IDE.



Visible are: file tree with integrated CVS interface (on the left), source code editor (with syntax highlighting), terminal window, build system manager (with automake backend)

15.8 Other tools

In this section let me point out two other tools that have significant impact on the development of HEP-related software already now, and will most probably have in the future: the ROOT package [60] and the PYTHON programming language[197]. The former is a comprehensive C++ framework for HEP-related programs, including a genuine C/C++ interpreter (CINT) and a rich class library for data analysis and visualisation. The latter is an Object-Oriented scripting language becoming more and more popular due to its extreme flexibility, and is likely to end the era of dominance of the PERL language. The main loop of the ATLAS analysis software (ATHENA) may already be constructed as a PYTHON script. PYTHON may also already be used with Root by means of PyRoot interface. PYTHON interfaces to other software packages (KDE, Qt, Apache web server scripting) are becoming available at a fast pace. I expect PYTHON and ROOT to become one of the most important software development technologies for HEP applications of the years to come.

Chapter 16

TAUOLA-PHOTOS-F

16.1 Introduction

The TAUOLA and PHOTOS programs are computing projects that have a rather long history. Written and maintained by well defined authors, they nonetheless migrated into a wide range of applications where they became ingredients of complicated simulation chains. As a consequence, a large number of different versions are currently in use. Usually, modifications required to adapt the package in the large simulation chain do not introduce important changes in the algorithm, but rather modify, for instance, the initialisation of physical constants, such as, for the example of TAUOLA, the form of the hadronic current in τ decays containing substantial amount of specific results from the distinct τ-lepton measurements. Typically, a dedicated interface for event data exchange following the conventions used by other components of the simulation chain needs to be provided as well. Having in mind the need for conservation of the rich physics content embedded in the (private or semi-public) versions of generators used by numerous experimental collaborations, and the need for development of the projects, a project capable of maintaining the versions of the code and providing a solid code repository appears to be indispensable. The task of organising and management of parallel versions of codes seems to be trivial to resolve using the object-oriented techniques, yet it may become troublesome in the case of FORTRAN codes, where programmers need to struggle with limited namespace and language functionality (e.g. it is impossible to pass the address of a function as a parameter). This kind of situation was encountered by the authors of the TAUOLA and PHOTOS generators; the TAUOLA-PHOTOS-F environment, presented in this chapter, is one of possible ways of resolving the inconveniencies. The package, which allows to construct specific versions of TAUOLA and PHOTOS, is dedicated for software librarians and advanced users of these packages. It consists of a code repository and a set of auxiliary utilities which allow to prepare versions of the FORTRAN source codes as published and used by experimental collaborations nowadays. At the same time, the repository is prepared to add the new versions of the codes, developed with the new collaborations right now. The first version of the TAUOLA-PHOTOS-F environment was documented in [154]; it was superseded by version *II*, documented in (recently updated and published) article[109].

The structure of TAUOLA-PHOTOS-F, being a form of software-management tool, is inspired by the requirements of the software used by CLEO [198], the experimental collaboration which was analysing the τ-lepton data for the longest time. The version of the code of TAUOLA as used in CLEO migrated to other collaborations, mainly Belle, but also BaBar and LHCb (through Fermilab) by means of the

QQ package [139]. These experimental collaboration are interested in performing comparisons of their results with the ones collected by CLEO, and use the software of the CLEO collaboration to perform such comparisons. Because of that, a version of the TAUOLA generator with an interface to CLEO software infrastructure is of interest - it is a matter of organisation of software and people collaborations. Apart from the one used by CLEO, two other TAUOLA interfaces are in wide use: the interface to the environment of the KKMC[55] and KORALB[171, 199] generators, and the TAUOLA Universal Interface, described in Chapter14. Similarly, for PHOTOS, various versions of the code that differ in the event generator being used and in the interface to the event record (format of HEPEVT) need to be maintained.

There are three mainstream physics initialisations for TAUOLA actually available in the main version of the package, and two options for 4-pion formfactors are provided as well. The new versions are being prepared upon request from B-physics collaborations (mainly Belle and BaBar) as a new option (TAUOLA-BBB subproject). All the options, initialisations, and interfacing modes may be freely combined in TAUOLA-PHOTOS-F to produce the desired version of the generator's source distribution directories, containing ready-to-compile FORTRAN codes of the generators and examples. The package also resolves the problem of evolving programming tools such as changing options for compilers and preprocessors and versions of the operating systems. Even though a standard Makefile-based build system was used by TAUOLA and PHOTOS distribution, the evolution of tools, platforms and hence differences in the requirements for the tools' options often rendered these build system unusable before. The solution, described further in Section 16.3, initiated further studies of usability of available build systems, which were reported shortly in Section 15.2.

The TAUOLA-PHOTOS-F package was initially accompanied by MC-TESTER (described in length in Chapter 17) tool for automatised comparison tests of versions of TAUOLA. MC-TESTER evolved as a separate project [200], yet the relics of its origin are still visible - is uses two versions of TAUOLA codes (with CLEO and ALEPH initialisations) as one of the available examples of its use.

The development of the version management and code repository functionalities of TAUOLA-PHOTOS-F and the validation/testing MC-TESTER tool were indispensable intermediate steps to the ambitious project of translations of the TAUOLA and PHOTOS generators to C++. The availability of MC-TESTER will allow to perform such translation while assuring the consistency of physics content of the generators; whereas TAUOLA-PHOTOS-F will provide a reliable source of reference (FORTRAN) implementations. We foresee the translation of the TAUOLA and PHOTOS to C++ in not so far future. The use of Object-Oriented techniques, such as polymorphism and inheritance will certainly be considered to achieve the extendability of TAUOLA (e.g. the implementation of custom form-factors and interfaces might be naturally separated, from the main code). The actual decision to start the process of translation depends on the need for C++ implementations of the TAUOLA and PHOTOS generators expressed by experimental collaborations; so far, it was not expressed.

16.2 Selection of the options for the code

To prepare the requested version of code, the standard UNIX "cpp" (C language preprocessor) command is used. Based on the options passed as arguments, it includes or excludes the appropriate variants of the "meta-source" code kept in the repository, producing a version of code that is ready to be compiled. To achieve that, the original source codes needed to be split to fragments, to separate the common code and the variants. The results, refereed to as "meta-sources" are not

to be used directly; instead, the TAUOLA-PHOTOS-F needs to be used to prepare a usable, "export" version of the `tauola/` and `photos/` directories, which in turn may be installed by the librarian into the collaboration software directory. The process of code preparation is started by executing the "make" command with an option that specifies requested version of the code, in the `TAUOLA-F/` and `PHOTOS-F/` directories.

Specifying the HEPEVT event record option In the `PHOTOS-F/` directory, the `make` command may be executed with the following options, in order to prepare the versions of PHOTOS and TAUOLA Universal Interface with certain format of the HEPEVT commom block:

- `KK-all` – for KK Monte Carlo (older version)
- `2kD-all` – arrays dimension 2000, double precision
- `4kD-all` – arrays dimension 4000, double precision
- `2kR-all` – arrays dimension 2000, single precision
- `10kD-all` – arrays dimension 10000, double precision

Specifying the interface and the initialisation for TAUOLA For TAUOLA, the following options for physical initialisation may be specified in the `TAUOLA-F/` directory:

- CPC (default) – the standard version of TAUOLA, as published in Computer Physics Communications Program Library, as published in [28], with improvements from Ref.[201];
- CLEO – initialisation from the CLEO collaboration [179]
- ALEPH – initialisation from the ALEPH collaboration [180]

In parallel to the source code of TAUOLA, the tool produces interfaces to various MC generators:

(1) Old demo program as in published version [28];

(2) Interface to `KKMC` [55];

(3) New universal interface using `HEPEVT` common block including spin effects in the elementary $Z/\gamma^* \to \tau^+\tau^-$ process [96].

(4) The extended version of the standard universal interface including transverse spin effects in the $H/A \to \tau^+\tau^-$ decay [172, 173, 174, 95].

The various options for event generator and the format of the HEPEVT event record, as specified in `PHOTOS-F/`, are also taken into account.

Apart from the three options for physical initialisation, there exist a possibility to introduce the parallel 4-pion channels, based on measurements and calculations of the Novosibirsk group [202] and Karlsruhe group [203, 201]. These 4-pion parameterisations may be introduced to one of the three versions mentioned above: CPC, CLEO or ALEPH. A dedicated mechanism, "tauola-factory" was implemented for that purpose: the complete meta-code of `TAUOLA-F/` is built from the fragments.

The very same mechanism was used to extend the TAUOLA-PHOTOS-F environment for maintenance of the evolving version the TAUOLA code, being implemented in collaboration with Belle and BaBar collaborations. Thanks to such organisation, all the code can be kept within a single package, and any version - either historical or a new experimental, with any interface may easily be reconstructed.

16.3 Build system organisation

The TAUOLA and PHOTOS projects survived many minor and major revolutions in the computing technology during the lifespan: the large "mainframe" computers , on which the programs were initially executed, were firstly replaced by powerful workstation; later, the dynamic development of the Open Source software, in particular the GNU/Linux, connected with rapid growth of commodity PC computers made it possible to run the simulations on typical office-grade computers or even laptops. The hardware architectures, operating systems, and software development tools, including the compiler systems were evolving in parallel at a rapid pace. Despite the fact the `make`-based software build systems are a strong standard (OpenGroup/POSIX)[184] which foundations remain almost unchanged, the tools driven by `make` **did** change throughout the years: the way the binary executable programs are organised in libraries changed when the dynamically-loaded libraries became a common element of all operating systems, and required changes in the sequence of the tools being used (linkers, archive managers) and the parameters that needed to be passed to the tools such as compilers, linkers and preprocessors. Even though the GNU Compiler Collection (gcc) System [204] is the most widespread compiler standard in Linux-based world of scientific computations, the problem with portability of the code still exists. Even within a single hardware architecture (such as i386), and operating system (GNU/Linux), and OS distribution (CERN Scientific Linux), the problem with portability remains: multiple choices of the version of the compiler are possible: apart from gcc, other commercial compilers may be employed. The gcc itself offers a variety of versions (gcc 2.95.2, 2.96, 3.2, 3.4, 4.0 being currently used on Linux), often not compatible with each other where it comes to compiling the C++ code.

The creation of the TAUOLA-PHOTOS-F package was therefore an opportunity to review the build system of the TAUOLA and PHOTOS packages, unify and clarify it, and introduce mechanism that allowed for better portability of the projects. The development of the system which is currently present in the TAUOLA-PHOTOS-F package was in fact the beginning of a wider study of software building tools, which is described briefly in Section 15.2.

The build system of the TAUOLA-PHOTOS-F is based on the separation of platform-independent parts of the *Makefiles*, such as targets and dependencies, from platform-dependent parts: the definitions of the actual tools (compilers, linkers) and their options. The platform-dependent part is stored in a single file: `platform/make.inc` , which is included by all *Makefiles* in the project. This way, any change, such as settings of the compiler, may be performed globally in the project, without the need to edit all the *Makefiles* that may be affected. It also allows to significantly simplify the actual *Makefiles*, which does not need to contain the technical details of compiler settings embedded in them. The portability of the project is also enabled: a set of ready-to-use versions of the `platform/make.inc` files, for various hardware/compiler platform may be prepared, tested, and distributed together with the project, or added later. We found this kind of organisation of the build system robust and flexible for our needs.

16.4 Issues related to distinct event record schemes

The algorithms employed by the TAUOLA and PHOTOS generators assume the tree-structure of the event record encoded properly in the standard `HEPEVT` event record. In general, however, that is not the case - the authors of generators choose to overload the (too rigid) HEPEVT structure with additional data and thus violate its tree-structure. Nevertheless, the programs were shown to work properly with

the extension of the standard of the type as originating from PYTHIA [34] with MSTP(128)=0,1,2. Necessary adjustments turned out to be non-trivial and in fact correctness of their action cannot be guaranteed in the general case [205]. The case of HERWIG [177] turned out to be even more complex. Even though in this case the HEPEVT common block is used as standard event record, the mother–daughter pointers are nonetheless used in a different way. As a consequence, events generated by HERWIG are far from the design format of the HEPEVT event record. It was possible to tune the working of the TAUOLA universal interface to the requirements of operating with HERWIG; however, the interfacing of PHOTOS turned out to be more difficult. The topic of interfacing PHOTOS and TAUOLA with event records is also covered in Chapters 13 and 14.

Chapter 17

MC-TESTER

In the phenomenology of high-energy physics, the question of establishing uncertainties for theoretical predictions used in the interpretation of the experimental data is of high importance. For example, at the time of the experiments at LEP several workshops (see [206]) were devoted to this question.

During the development of the TAUOLA [28, 29, 27] generator, a set of tests that performed comparisons of results produced by two versions of the program. It was determined that a test that compares all distributions of invariant masses built from the decay products of the τ lepton (in a particular decay channel) gives a valuable and quite complete answer to a question of compatibility of the two versions of the generator. The test is sensitive to a broad range of problems encountered in the event generator codes, and is capable of indicating the nature of many of them. Moreover, the use of an observable with physical meaning make it suitable to test (compare) the physics models implemented in the generators; comparisons with experimental data could also be performed with this kind of test, and provide valuable physical information.

While working on the TAUOLA-PHOTOS-F package (see Chapter 16), the creation of a semi-automated tool for comparisons of versions of TAUOLA code, based on the concept of this test was proposed. The new tool, MC-TESTER, which was initially a part of the TAUOLA-PHOTOS-F package, evolved to an independent projects with the area of use much wider than initially expected: universal tool performing semi-automatic comparison tests on the predictions of Monte Carlo event generators. Even though initially limited only to the particle decay processes, it was easily extended to work for $2 \to n$ processes, as requested in the Linear Collider Workshop [207].

MC-TESTER was described in length in [200], and refereed to in a number of other papers, therefore I will not present its detailed technical description here. Instead, a brief reminder of the main concepts and mechanisms of MC-TESTER will be presented. I will also focus on the details not discussed before, which were exploited in the precision tests of the PHOTOS algorithm, described further in Chapter 18. It was the availability of MC-TESTER that enabled the possibility of performing such tests.

The design, development and application of MC-TESTER to various physical problems was one of the main areas of my scientific activities, related to HEP simulations, throughout recent years. The code and documentation is available on my web-page [208].

17.1 Concepts

At the most abstract level, MC-TESTER is a tool providing an answer to the question of statistical compatibility of distributions of some variable. This is a well-known problem that can be discussed at the ground of statistical mathematics; data analysis in the high-energy physics experiment often need to employ this, standard, methodology to obtain the ultimate results. I do not intend to discuss this topic here. I will point out that the analyses are typically performed on histogrammes of values, by means of dedicated software tools, such as ROOT or PAW.

The concept of the test, mentioned already above, is to perform comparisons of distributions of all possible invariant masses that can be constructed from decay products in some process. In such approach hundreds of distributions may need to be compared: an automated process of extracting the distributions, performing the comparisons, and presenting the results in a readable, way would be of value. MC-TESTER provides such solution, based on a two-step process: data collection ("generation step") and comparison of distributions ("analysis step"). By providing an easy-to-use functions that automatically extract the required distributions from the events produced by event generators, MC-TESTER minimises the ambiguities in the definition of the distributions and possibility of making an error. The analysis-related part presents the results in a visualised, easy-to-browse form of a "booklet" of plots of histograms, makes it possible to quickly identify and classify the discrepancies between the distributions. Ultimately, the use of ROOT package to store the distributions (in form of histograms) and perform the comparisons make it possible to extend the package and tailor it easily to specific needs.

When used in its standard form, MC-TESTER compares particle decays generated by two Monte Carlo event generators. First, appropriate data (invariant mass distributions) needs to be collected from a run of every tested program. To this end the libraries of `MC-TESTER` have to be loaded, a testing routine called, and an identifier of the particle to be analysed has to be provided. Event after event, `MC-TESTER` will read data from consecutive event records (such as `HEPEVT`) and look for decays of the tested particle. Once found, a decay channel is identified from its decay products. On the first occurrence of the particular decay channel, it is added to a list of found decay modes; all histograms for invariant masses that can be formed of decay products are initialised and filled for the first time. For later occurrences, appropriate histograms are simply filled. At the end of the run all information is stored to a (root) file.

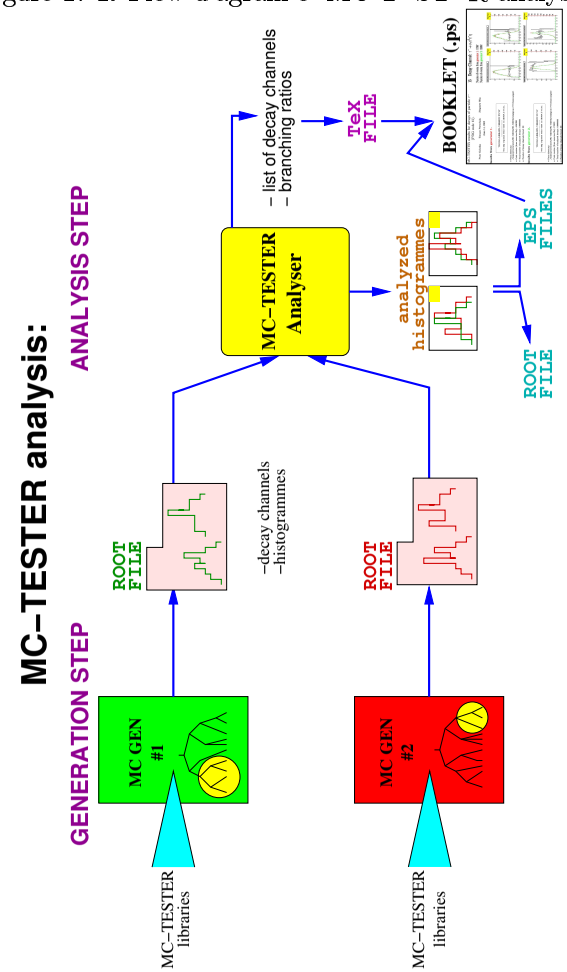
In the second step, the results obtained from the two generation runs from two Monte Carlo programs are compared and a booklet is created. It includes a table of decay modes found in the two runs with corresponding branching ratios. For matching decay channels (present in outputs from both generators), comparison plots are provided. For each invariant mass distribution, histograms from the two runs are plotted; their ratio (after normalisation) is also plotted in the same frame. The `Shape Difference Parameter` (`SDP`) is calculated and printed on the plot as well. The maximum of `SDP` over all plots for the given decay channel is printed in the table of decay modes as well.

The complete, two-stage process is presented in Figure 17.1.

17.2 Generation step

To perform the generation step, runs of the event generator programs, instrumented with calls to MC-TESTER library need to be performed. It is sufficient to link the generator with the MC-TESTER libraries (and ROOT libraries), as described i.e. in [200], and insert a call to the MCTEST() routine (in C++ or in FORTRAN), that

Figure 17.1: Flow-diagram of MC-TESTER analysis.



should be executed after a new event is generated. The remaining few parameters (the type of the particle the decays of which are analysed, the size and ranges of histograms, descriptive text, optional event modification steps) may be specified each time at the run-time in the configuration file SETUP.C, which is a C++ macro executed by ROOT interpreter. The possibility of performing custom event-record modifications every time a new event is generated was exploited heavily in the PHOTOS high precision tests (see Chapter 18). The list of MC-TESTER parameters that can be controlled from SETUP.C is presented in the Appendix of [200]. The results of a single run of the generator are stored in the `mc-tester.root` file.

17.3 Analysis step

The results of generation step stored in the data files `mc-tester.root`, referred in the previous section, are used to produce a booklet (Fig.17.2 and Fig.17.3) - a final results of the `MC-TESTER` action. The analysis is performed in the dedicated `analyze/` directory, which holds the analysis environment of the MC-TESTER. The two `mc-tester.root` files produced by two compared versions of generators, need to be put in the `prod1/` and `prod2/` subdirectories, and the analysis step needs to be executed using the `make` command. Optional parameters, that control the analysis step, may again be specified in the SETUP.C C++ macro file. As a result, the complete booklet with results (file `tester.ps`) is produced; it consists of:

- The title page giving details of performed tests, ID of a tested particle, names and a short description of the two programs under test, numbers of generated events and numbers of decay channels found in the two runs .
- The table of decay channels found with branching ratios from the two runs and maximum of the `Shape Difference Parameter` for all histograms defined for the channel;(see Fig. 17.2(b)).
- The table of content indicating a page number where plots for a given channel start.
- Plots of histograms of all invariant masses for all matching decay channels found in the two runs; (see Fig. 17.3(a)).

Along with the booklet, a `ROOT` output file `mc-results.root` is generated. It may be analysed further, by the advanced user. A complete set of all plots of the booklet (in the *.eps* format) is also stored in subdirectory `booklet/` .

17.3.1 Description of a single plot

The plots such as in Fig.17.3(b) are the main part of the booklet. They allow for the visual estimation how a certain mass distribution compares between the two programs. There are three histograms in each plot. Two histograms with the distributions from the first and the second generator, are plotted in red and green (or darker/lighter grey), and a histogram of the ratio of the two (in black). The red and green colours are used consistently throughout the booklet to refer to the generators.

The left axis on the plot refers to the histogram of the ratio. This histogram is obtained from the division of the normalised histograms, therefore if the shapes of the two mass histograms match (even though they differ in statistic), the division histogram will be up to statistical fluctuations matching flat distribution at 1.0 . The division histogram is the first, visual comparison test.

MC-TESTER results for decays of particle τ^- (PDG code 15).

Piotr Golonka Tomasz Pierzchala Zbigniew Was

June 14, 2002

Results from generator 1.

TAUOLA LIBRARY: VERSION XX.YY
.....
You may replace this text in SETUP.C file.

- From directory: /home/piters/work/MC-TESTER/MC-TESTER/examples-F77/tauola/gen1
- Code version (from version file): ALEPH
- Total number of analyzed decays: 10000000
- Number of decay channels found: 30

Results from generator 2.

TAUOLA LIBRARY: VERSION UU.VV
.....
You may replace this text in SETUP.C file.

- From directory: /home/piters/work/MC-TESTER/MC-TESTER/examples-F77/tauola/gen2
- Code version (from version file): CLEO
- Total number of analyzed decays: 10000000
- Number of decay channels found: 30

(a) The first page of analysis booklet.

Found decay modes:

Decay channel	Branching Ratio ± Rough Errors		Max. shape dif. param.
	Generator #1	Generator #2	
$\tau^- \to \nu_\tau \pi^0 \pi^-$	25.3683 ± 0.0159%	25.3085 ± 0.0159%	0.04375
$\tau^- \to e^- \widetilde{\nu}_e \nu_\tau$	17.8479 ± 0.0134%	18.1093 ± 0.0135%	0.00000
$\tau^- \to \mu^- \widetilde{\nu}_\mu \nu_\tau$	17.3866 ± 0.0132%	17.6326 ± 0.0133%	0.00000
$\tau^- \to \nu_\tau \pi^-$	11.0768 ± 0.0105%	11.1765 ± 0.0106%	0.00000
$\tau^- \to \nu_\tau \pi^0 \pi^0 \pi^-$	9.1865 ± 0.0096%	9.1171 ± 0.0095%	0.09413
$\tau^- \to \nu_\tau \pi^+ \pi^- \pi^-$	8.9837 ± 0.0095%	8.8828 ± 0.0094%	0.09368
$\tau^- \to \nu_\tau \pi^0 \pi^+ \pi^- \pi^-$	4.2973 ± 0.0066%	4.5319 ± 0.0067%	0.30310
$\tau^- \to \nu_\tau \pi^0 \pi^0 \pi^0 \pi^-$	1.0765 ± 0.0033%	1.0090 ± 0.0032%	0.00724
$\tau^- \to \nu_\tau K^-$	0.7202 ± 0.0027%	0.7138 ± 0.0027%	0.00000
$\tau^- \to \nu_\tau \pi^0 \pi^0 \pi^+ \pi^- \pi^-$	0.4990 ± 0.0022%	0.0897 ± 0.0009%	0.00000
$\tau^- \to \nu_\tau \pi^0 K^-$	0.4785 ± 0.0022%	0.4617 ± 0.0021%	0.00000
$\tau^- \to \nu_\tau K_L^0 \pi^-$	0.4624 ± 0.0022%	0.4444 ± 0.0021%	0.00000
$\tau^- \to \nu_\tau \pi^- K_S^0$	0.4610 ± 0.0021%	0.4449 ± 0.0021%	0.00000
$\tau^- \to \nu_\tau \pi^+ \pi^- K^-$	0.3902 ± 0.0020%	0.5051 ± 0.0022%	0.52330
$\tau^- \to \nu_\tau \pi^0 \pi^- \eta$	0.1707 ± 0.0013%	0.1696 ± 0.0013%	0.00000
$\tau^- \to \nu_\tau \pi^- K^+ K^-$	0.1704 ± 0.0013%	0.1509 ± 0.0012%	0.07360
$\tau^- \to \nu_\tau \pi^0 K_L^0 \pi^-$	0.1605 ± 0.0013%	0.2745 ± 0.0017%	0.92850
$\tau^- \to \nu_\tau \pi^0 \pi^- K_S^0$	0.1592 ± 0.0013%	0.2734 ± 0.0017%	0.93657
$\tau^- \to \nu_\tau \gamma \pi^0 \pi^-$	0.1559 ± 0.0012%	0.1303 ± 0.0011%	0.00000
$\tau^- \to \nu_\tau K_L^0 \pi^- K_S^0$	0.1510 ± 0.0012%	0.0763 ± 0.0009%	0.00270
$\tau^- \to \nu_\tau K_L^0 K^-$	0.1289 ± 0.0011%	0.0508 ± 0.0007%	0.00000
$\tau^- \to \nu_\tau K_S^0 K^-$	0.1287 ± 0.0011%	0.0507 ± 0.0007%	0.00000
$\tau^- \to \nu_\tau \pi^0 \pi^0 \pi^0 \pi^+ \pi^- \pi^-$	0.1094 ± 0.0010%	0.0506 ± 0.0007%	0.00000
$\tau^- \to \nu_\tau \pi^+ \pi^+ \pi^- \pi^- \pi^-$	0.0803 ± 0.0009%	0.0401 ± 0.0006%	0.00000
$\tau^- \to \nu_\tau \pi^0 \pi^0 K^-$	0.0792 ± 0.0009%	0.0504 ± 0.0007%	0.29190
$\tau^- \to \nu_\tau K_L^0 K_L^0 \pi^-$	0.0760 ± 0.0009%	0.0372 ± 0.0006%	0.00854
$\tau^- \to \nu_\tau \pi^- K_S^0 K_S^0$	0.0756 ± 0.0009%	0.0378 ± 0.0006%	0.01189
$\tau^- \to \nu_\tau \pi^0 K_L^0 K^-$	0.0507 ± 0.0007%	0.0763 ± 0.0009%	0.85321
$\tau^- \to \nu_\tau \pi^0 K_S^0 K^-$	0.0498 ± 0.0007%	0.0746 ± 0.0009%	0.87506
$\tau^- \to \nu_\tau \pi^0 \pi^+ \pi^+ \pi^- \pi^- \pi^-$	0.0186 ± 0.0004%	0.0293 ± 0.0005%	0.00000

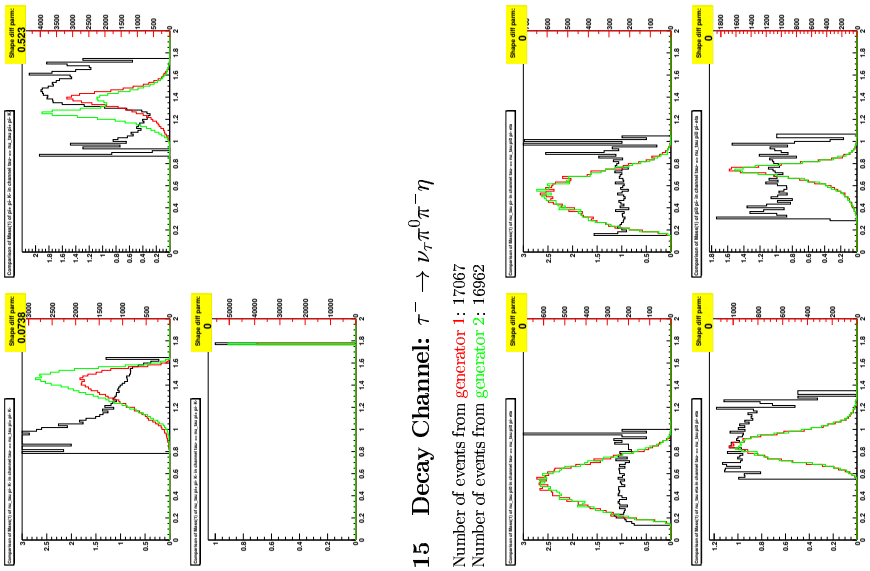
Similarity coefficients: T1=1.881148 , T2=4.510389

(b) The table of found decay channels.

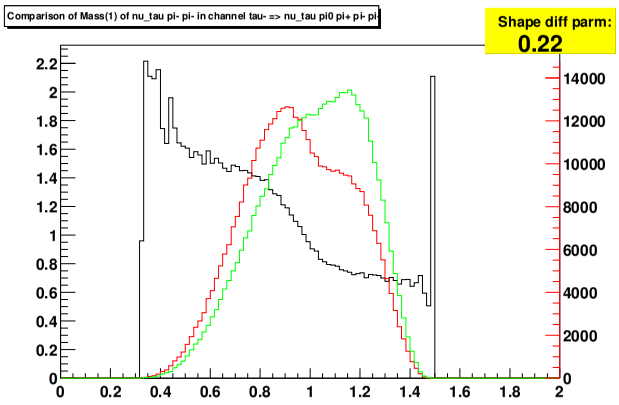
At the bottom of the table (b), the values of the T_1, T_2 coefficients, quantifying the difference in all decay channels combined, are printed.

Figure 17.2: Example of booklet's informational pages produced at analysis step

Figure 17.3: Examples of plots produced at analysis step.



(a) The example of a page with histograms.



(b) The example of a histogram.

The right axis of the plot refers to the contents of the mass histograms, therefore represents the number of entries in every single mass bin. These histograms are not normalised, and large differences in statistical samples will immediately show up. One will observe that the histogram with smaller number of entries will occupy small (negligible) part of the plot. The range on the right axis is chosen such, that both histograms will show up in full. In the top-right corner of the plot (on yellow background), SDP (the Shape Difference Parameter) is printed.

The total number of events found for a given channel in the runs of the two generators is printed in the booklet, just at the beginning of a section including plots for a channel (see Fig. 17.3(a)).

17.4 Shape Difference Parameter calculation algorithms

An analysis performed by MC-TESTER consists of comparisons of distributions in every possible invariant mass constructed over every (sub-)set of decay products of every decay channel. The number of histograms that need to be compared may reach thousands[1]. It is therefore convenient to define a single measure (called the Shape Difference Parameter (SDP) here), measuring the difference between the same distributions from the two programs. Later, for every analysed channel, the maximum of SDP over all pairs of distributions, from the two programs can be found and printed in the review table of MC-TESTER (in the last column in the Fig.17.2(b)).

Depending on the user's particular need, the choice of an algorithm calculating the value of the SDP may be different. Also, shapes of the distributions (which depends on the decaying channel) may affect the choice. On the other hand, the main purpose of SDP is to draw user attention to decay channel where the differences of predictions from the two programs differ maximally. That is why the choice of SDP in many cases should not be of prime importance. At present, three options are provided, which were found useful in work on the TAUOLA Monte Carlo. For unsatisfied user, they may serve, at least, as examples to develop a new one. In the following I will list the available tests (the definitions and interpretation of the algorithms for these tests may be found in [200] and [3].):

MCTest01

This test determines the size of the area under the two compared, normalised to unity histograms, which is not simultaneously under the two of them. Estimates of the statistical fluctuations are subtracted bin-by-bin (in the limit of 3 standard deviations). The test gives SDP equal to 0 when the compared histograms are statistically identical, and SDP equals 2 when the histograms are completely disjoint.

MCTest02

The aim of this test is to measure, how far from a constant is a ratio of two histograms. The difference of this test with respect to the previous one is, that it weights equally all bins, and is not focused on the most populated bins as in the previous case. The test returns SDP=0 when the compared histograms are statistically identical. In general case, in the limit of infinitely large samples, it returns the surface between two lines: the ratio of the two (normalised) histograms (see e.g. the black line and the left scale on Fig.17.3(d)) and the constant line at 1.

[1] Already for the decays of the τ lepton as implemented in TAUOLA, there are 30 distinct decay modes; for a typical (5 body) decay channel the algorithm defines 26 distinct distributions,

`MCTest03`

In some applications, such as checks of proper installation into broader environment of a code for the particle decay, we may be interested if indeed after installation, all distributions remained unchanged. For that purpose Kolmogorov compatibility test is suitable. It calculates a probability p that the two histograms have the same shape. As this test is implemented and documented as a standard function of `ROOT` [209, 210], we will use it for the time being, and skip a description as well. We simply take `SDP`$=1-p$. Test returns `SDP`$=0$ if two distributions are identical, in other cases it returns the probability of being different.

The actual test used during the analysis may be defined in the `SETUP.C` of the directory `analyze/` . The users may define their own tests following the examples of the aforementioned tests.

17.5 Access to event record data

At the generation step, MC-TESTER needs to extract the invariant mass distributions from the data contained in the event records. The unified access to various (FORTRAN) event record standards from C++ code is provided by HEPEventLib "compatibility layer", already discussed in Section 11.6.9. The main concepts used in the HEPEventLib prove to work efficiently: the iterators to search and navigate through the event record, and transparent access to various event records. MC-TESTER was used successfully with HEPEVT, LUJETS, PYJETS, and HERWIG's variant of data-overloaded event record. In the latter case, the problem of relation-pointers overload was eliminated transparently in the HEPEventLib!

17.6 Extensions

The `MC-TESTER` has been proposed as the tool for the Linear Collider Workshop [207] for comparisons of Monte Carlo predictions for $2 \to n$; (n=4,6,8) body generators. Some changes were necessary.

First, the possibility to process the program input was introduced. The scattering event of the form $p_1 p_2 \to q_1 ... q_n$ can be replaced by the decay-like chain : $P \to p_2 q_1 ... q_n$, where $P = p_1 + 2 \times p_2$. The first incoming particle (of momentum p_1) is replaced by the P, the "mother" of all other particles, including second beam (p_2). One obtains the event record with decay tree-like structure, i.e. standard event record for `MC-TESTER` use. The flexibility of the run-time root-macro interpretation mechanism employed by MC-TESTER allowed to implement the extension as a simple, local event record modification.

The second extension prepared for the Linear Collider Workshop allows to quantify the difference of predictions of two programs for $p_1 p_2$ (or p) $\to$ *anything* process. This is done with the help of two real numbers, T_1, T_2 (printed under the table of decay channels - page 2 of the booklet), which can be later analysed if comparison of more than two programs is performed.

The first number T_1 represents the difference due to branching ratios from program A and B. The second one consists of the weighted sum of maximal shape difference parameter calculated for all channels. The weight is given by the average of the branching ratios (calculated for i channel). For more detailed definition please refer to [200].

Part VI

Validations

Chapter 18

Photos as a precision tool

From the point of view of the precision of the obtained results, the PHOTOS algorithm may work in three regimes: (1) as a "crude-level" tool for bremsstrahlung generation in decays of any particle or resonance, (2) as a precision tool for dedicated decay channels (at present Z and W decays), (3) as a precision tool incorporating matrix-element calculations (not exploited so far). At a "crude level", only the leading-log soft-photon parts of the matrix element are included in the PHOTOS algorithm[1]. PHOTOS is most often used as a general-purpose tool, thus working at the "crude level" of precision. The precision of the results cannot be guaranteed in this working regime; a typical example of use could, however, be the generation of full bremsstrahlung phase-space coverage for acceptance studies only. Appropriate comparisons with matrix-element calculations or experimental data would have to be performed to determine the actual precision. For the particular decay channel, the PHOTOS algorithm could then be upgraded to regime (2) or (3), depending on the form of necessary compensating weight. At present, the algorithm employs a number of improvements (with the help of the correcting weight) and takes into account such effects as the threshold terms for fermions and the impact of the spin of emitting particle. Other effects (e.g. the impact of spin of a decaying particle or the influence of spin on interference terms) are not at present taken into account.

The impact of the approximations and missing terms on the precision of PHOTOS, when used in regime (2), that is for W and Z decays, is the main subject of this chapter; the results presented here have also been published in [153].

18.1 Test definition

Our approach to the study of precision of the PHOTOS algorithm presented in this paper is based on numerical comparisons of results obtained from PHOTOS with respect to the results produced by other *reference* Monte Carlo generators. The reference generators that we used employ formulae based on the full (i.e. non-approximated) fixed-order matrix-element calculations with or without exponentiation; their precision level is well established, both by theoretical considerations and by comparisons with experimental data. The advantage of such an approach is obvious: it allows us not only to check the theoretical precision of the approximation used, but also verifies that there are no accidental errors in the actual computing code.

[1] It can be estimated that the physical uncertainty of the PHOTOS algorithm, with double emission and working on the decay of particle P into a charged particle ch and the neutral system Y_i, is then not smaller than $\frac{\alpha}{\pi}$ or $(\frac{\alpha}{\pi}\log\frac{m_P^2}{m_{ch}^2})^3$, whichever is bigger.

We have tested PHOTOS against a limited number of event generators and for a limited number of processes. The choice of processes was dictated not only by physic interests, but also by the availability of Monte Carlo generators with well-controlled and established precision. The following processes were studied:

- $Z^0 \to \mu^-\mu^+$: for this process the LEP era generators KORALZ [160, 211] and KKMC [55] could be used. Note that these programs agreed well with practically all experimental data of the LEP measurements. The KKMC generator is based on $O(\alpha^2)$ ME calculations with spin amplitudes technique and exponentiation. It can also be used in $O(\alpha)$ ME exponentiated mode. In the case of KORALZ, generation at first-order matrix element and no exponentiation were available as well.

- $W^+ \to \mu^+\nu_\mu(\gamma)$: for this process the WINHAC [212] generator is available for multiple photon generation in decay. It is based on first-order matrix element and exponentiation. Comparison of PHOTOS with a first-order matrix-element generator without exponentiation can be found in [167]; we will not repeat it here.

- $H \to \mu^+\mu^-(\gamma)$: comparison of PHOTOS with a first-order matrix-element generator without exponentiation can be found in [170]; we will not repeat it here.

- for leptonic τ-decays, complete QED first-order generation with the TAUOLA [29] generator was available for tests; TAUOLA was widely used and compared successfully with LEP and CLEO data [213, 214].

The event generators and physical initializations mentioned above will be referred to as "reference generators". They were used to produce reference results used in comparisons with PHOTOS.

The testbed constructed from the event generators discussed above had one very important feature: it allowed a test of PHOTOS against several algorithms that differ in physical content and precision. Tests started from an exact $O(\alpha)$ matrix element for single-photon emission (non-exponentiated KORALZ and TAUOLA) compared with a single-photon version of PHOTOS), and went on to multiple-photon generators (exponentiated versions of KKMC, KORALZ and WINHAC) compared with triple-, quartic- and exponentiated multiple-photon modes of PHOTOS. To operate, PHOTOS needs events produced by a "host" generator as an input. We exploited the possibility of deactivating the bremsstrahlung generation in reference generators, to turn them into QED Born-level "host" generators.

Let us now define the method for automated tests that were used to obtain the results presented later in this paper. To facilitate the systematic comparisons, we adopted MC-TESTER, described in Chapter 17. The testing approach implemented in MC-TESTER was already very useful in the case of validation of TAUOLA package; however, for the purpose discussed here it required further adaptation. The problem was in consistent treatment of the arbitrary number of final-state QED bremsstrahlung photons that might be present in the event, the ambiguity caused by infrared singularities of QED, which is handled differently by the various Monte Carlo programs.

Our aim was to develop a technique where comparisons would make physical sense, would be automatic and independent of the way infrared singularities are regularised in particular generators. For our analysis, we defined zero-, one-, and two-photon topologies in the following way: we called the event "zero photon" if there was no photon of energy (in a decaying particle rest-frame) larger than the parameter $E_{\rm test}$. The "one-photon" event had to have one (and only one) photon

of energy larger than $E_{\rm test}$. If there were more than one such photons, we called it a "two-photon" event. If there were more than two photons of energy larger than $E_{\rm test}$, we considered only the two most energetic ones, and treated the remaining ones as if they had not passed the $E_{\rm test}$ threshold. For all the photons that did not pass the $E_{\rm test}$ threshold we performed an operation inspired by leading-log logic: we added their four-momenta to the momenta of outgoing fermions of smaller angular separation.

We defined two variants for this test definition: *test1* and *test2*. The *test2* was exactly as explained above. In *test1*, only one photon (the most energetic one) could be accepted[2]. The free parameter of the test: $E_{\rm test}$ was adjusted for each process so that the results had physical sense. For the purpose of presentation of the results we decided to extract the most vital information: the branching ratios for events with 0, 1 and 2 photons (0 and 1 if *test1* was used) and the maximum value of SDP throughout all the plots in a channel. Complete results with summary tables and all analysis booklets are available on the web [168]. The tests were based on high-statistics runs (10^8 non-weighted events) for all generators.

Finally, let us stress that an essential preliminary step for the tests presented here was to assure the numerical stability of PHOTOS (described in Section 13.3). As a consequence, we could determine that PHOTOS may be used for processes at very high energies (i.e. at the scale of LHC energies and even at astrophysics ones).

18.2 Test results

The results presented here cover two issues: precision of the predictions obtained for the single-photon emission kernel and the convergence of the iterative solution. At first, in subsection 18.2.1, we will discuss the comparisons of a single-photon emission kernel implemented in PHOTOS with other genuine first-order generators. In subsection 18.2.2, we will discuss tests for multiple-photon generation. Finally, in subsection 18.2.3, we will discuss the applicability of PHOTOS at very high energies.

As mentioned before, we aim at comparing, in total, *six* versions of distinct Monte Carlo programs with five distinct modes of executing PHOTOS. Let us specify them here (including acronyms that will be used throughout the text):

- **KORALZ O(1)**: KORALZ generator [211] with $O(\alpha)$ ME for $Z^0 \to \mu^+\mu^-(\gamma)$, no exponentiation, single-photon emission
- **KORALZ** : KORALZ generator [211] with $O(\alpha^2)$ ME for $Z^0 \to \mu^+\mu^-(\gamma)$ and exponentiation (multiple-photon emission)
- **KKMC** : KKMC generator [55] with $O(\alpha^2)$ ME for $Z^0 \to \mu^+\mu^-(\gamma)$ and exponentiation at spin-amplitudes level
- **KKMC** $O(1)_{\rm EXP}$: KKMC generator [55] with $O(\alpha)$ ME for $Z^0 \to \mu^+\mu^-(\gamma)$ and exponentiation
- **TAUOLA** : TAUOLA generator [29] with $O(\alpha)$ ME for $\tau \to l\bar{\nu}_l\nu_\tau(\gamma)$ (single-photon emission)
- **WINHAC EXP** : WINHAC generator [212] with full $O(\alpha)$ ME for $W \to l\bar{\nu}_l(\gamma)$ and exponentiation
- **PHOTOS O(1)**: PHOTOS, no iteration

[2] If needed, this scheme could be generalised, for instance, to define *test3*, where up to three photons of energies above $E_{\rm test}$ could be accepted, leading to a fourth distinct decay channel.

- **PHOTOS O(2)**: PHOTOS algorithm iteration up to two times
- **PHOTOS O(3)**: PHOTOS algorithm iteration up to three times
- **PHOTOS O(4)**: PHOTOS algorithm iteration up to four times
- **PHOTOS EXP** : PHOTOS algorithm with exponentiation, multiple-photon generation

Before we proceed with a presentation of the results, let us, however, briefly describe the approach taken for this presentation and their numerical quantification:

- As described in section 18.1, we quantify the difference between results produced by two generators by calculating branching ratios for events with 0, 1 or 2 photons with energy above the $E_{\rm test}$ threshold, and the maximum values of the SDP parameter of all combinations of invariant masses in the specific channel (with 0, 1 or 2 photons) for a given comparison.
- In discussion of the results we will also often use a single value of the "overall difference", being either the maximum of differences in branching ratios or the maximum of products of SDP and corresponding branching ratios[3]. We then take the larger of the two.

Later in this section we present only the simplified summary tables and plots where the differences in distributions (quantified by the SDP parameter) are most significant. In the summary tables the first line (appearing in bold font) refers to the results produced by the reference generator, the others refer to results of runs of generators being tested.

18.2.1 Single-photon emission kernel

In this subsection we focus on the quality of the single-photon emission kernel in the PHOTOS algorithm. We perform a systematic comparison of PHOTOS with generators that implement $O(\alpha)$ matrix element for Z, W and τ decays.

The purpose of these tests was to assess the impact of the simplifications of the photon-emission kernel implemented in PHOTOS on the precision of its predictions (in the PHOTOS kernel the non-leading $O(\alpha)$ terms are omitted).

At first, let us present the comparisons made in the $Z \to \mu\mu(\gamma)$ channel. In fact, the original PHOTOS algorithm was created by gradual simplification of the exact ME formula for this process, so the universal emission kernel could have been identified. Therefore this process remains a benchmark for PHOTOS single-emission algorithm, its precision and design. The approximations are well understood and may in fact be restored for this particular case.

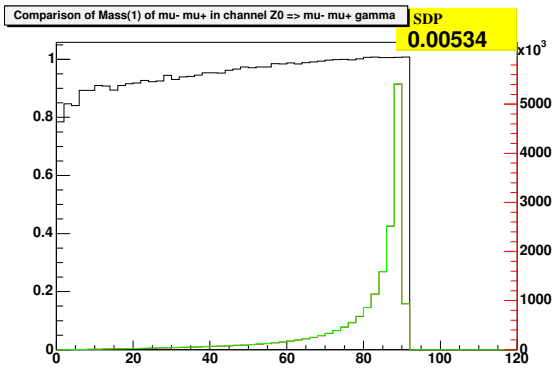
We compare PHOTOS with KORALZ Monte Carlo in single-photon emission mode. This test, which was already performed a long time ago (when PHOTOS was being developed), has been reproduced now, and confirmed that no accidental error has occurred since then. The results of this comparison are presented in Fig.18.1. One may notice that the difference in branching ratios for channel with and without photon are at the per mil level. The differences in distributions (the maximum value of SDP multiplied by branching ratios) indicate agreement better than per-mil: we have quantified the overall difference as 0.15%.

The dominant contribution to the difference originates from a very sparsely populated area of the phase space, where the invariant mass of the lepton system is

[3]For instance, in case of the table in Fig. 18.1 the difference in branching ratios is $0.82514 - 0.82362 = 0.00152$, the product SDP $\times$ BR $= 0.0053 * 0.176 = 0.00093$, so that the first value is taken as the overall difference for that test: 0.152%.

Figure 18.1: Predictions of KORALZ (with $O(\alpha)$ matrix-element and single-photon emission) are compared with predictions of PHOTOS (running in single-photon option) for the $Z^0 \to \mu^+\mu^-(\gamma)$ channel, $E_{\rm test} = 1.0$ GeV. The plot presents the distribution of invariant mass of $\mu^-\mu^+$ pair coming from KORALZ (in red, or darker-grey) and PHOTOS (in green, or lighter-grey); the black line is the ratio of the two normalised distributions. The red and green lines are hard to separate - they practically overlap.

GENERATOR	Branching ratio		Max SDP	
n photons $\to$	0	1	0	1
KORALZ O(1)	**0.82514**	**0.17486**		
PHOTOS O(1)	0.82362	0.17638	0	0.0053



small, and very hard photons were emitted. The discrepancy between the distributions does not exceed 20%, even in this corner of the phase space, though. It is visible in the slope of the black curve denoting the ratio of the two distributions. In this region of the phase space the approximations (with respect to $O(\alpha)$) used in PHOTOS are the largest. The differences are too small to justify the implementation of a channel-dependent correction weight.

For the case of W-boson decays, we address the reader to [167], where the comparison of PHOTOS and a $O(\alpha)$ ME generator SANC [215][4] for the $W \to \mu\bar{\nu}_m u(\gamma)$ process is given. This paper provides also the theoretical background for the W interference weight implemented in 2003 in PHOTOS 2.07. Presented tests indicate very good agreement between PHOTOS and SANC: up to a level of 1% (statistical error) in the majority of the areas of the phase space, within 5% for parts of distributions where collinear-induced logarithms are absent and 10% in the regions where only non-leading corrections contribute to the matrix element.

The paper [170] presents the comparison of various distributions from PHOTOS and SANC [215] generators for Higgs boson decays: an agreement at the level of 1% (statistical error) was found all over the phase space.

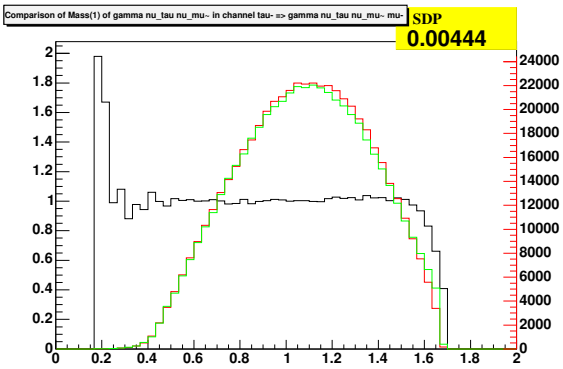
We consider the results of these tests, published in relatively recent papers [167, 170], as being relevant and complementary to the tests presented here. Therefore we do not repeat the comparison of WINHAC generator [212] and PHOTOS in single-photon emission mode.

We have however exploited the opportunity of having another comparison with full $O(\alpha)$ Matrix Element generator, that is TAUOLA [29] for leptonic τ decays. The

[4]SANC calculates complete one-loop amplitudes for the decays of on-shell vector bosons: W, Z or the Standard Model Higgs boson; an exact, single, real-photon emission matrix element is obtained that way.

Figure 18.2: Predictions for exact $O(\alpha)$ in TAUOLA are compared with single-photon emission of PHOTOS in the $\tau^- \to \mu^- \bar{\nu}_\mu \nu_\tau(\gamma)$ channel, $E_{\rm test} = 0.05$ GeV. The plot presents the distribution of invariant mass of the $\bar{\nu}_\mu \nu_\tau \gamma$ system coming from TAUOLA (in red, or darker-grey) and PHOTOS (in green, or lighter-grey); the black curve is the ratio of the two normalised distributions.

GENERATOR	Branching ratio		Max SDP	
n photons $\to$	0	1	0	1
TAUOLA	**0.98916**	**0.01084**		
PHOTOS O(1)	0.98927	0.01073	0	0.0044



results of comparison in $\tau^- \to \mu^- \bar{\nu}_\mu \nu_\tau(\gamma)$ decay channel are presented in Fig. 18.2. The overall difference was quantified as 0.11%. The agreement between PHOTOS and TAUOLA results is excellent. In the $\tau^- \to e^- \bar{\nu}_e \nu_\tau(\gamma)$ channel, the agreement is equally good.

18.2.2 Iterating emission kernels

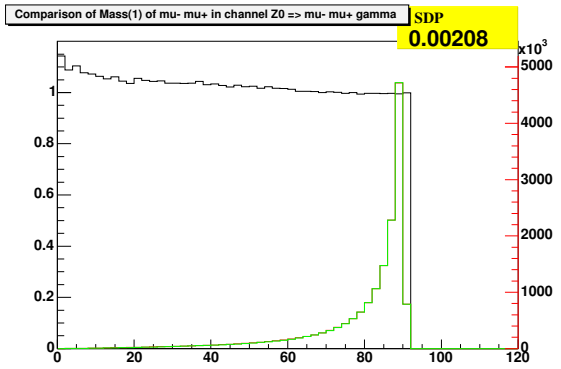
In this subsection we cover the comparisons of PHOTOS with other generators for multiple-photon generation. In multiple-photon mode, PHOTOS algorithm iterates the single-photon emission kernel, which precision was established in the previous subsection. The main questions to be answered by the group of tests presented here were the ones concerning the convergence of the PHOTOS iterative algorithm for photon emission to solutions implemented in other generators.

The KKMC Monte Carlo program [55] played an important role in multiple-photon comparisons: it is the only available Monte Carlo that implements a complete $O(\alpha^2)$ matrix element for two hard photon emission in the $Z \to \mu^+ \mu^- n(\gamma)$ channel. Because of its superior quality with respect to KORALZ [211] (which implements $O(\alpha^2)$ ME with more approximations) it was used as the reference generator throughout all tests in this channel. Before continuing with tests of PHOTOS let us first compare the KORALZ and KKMC Monte Carlo programs to assess the impact of the presence of second-order terms in the calculations. The results of comparisons performed using *test1* are presented in Fig. 18.3. The overall difference is below a per-mil level; however, with samples of 10^8 events the contribution from those $O(\alpha^2)$ terms that are missing in KORALZ is already visible. As will be shown later, the differences are in fact at the level of (and are comparable in shapes to) the differences observed between KORALZ and PHOTOS.

We start the tests of PHOTOS from comparisons for the benchmark decay $Z \to \mu^+ \mu^- n(\gamma)$. The complete summary of results is presented in Table 18.1

Figure 18.3: Comparison of predictions from KKMC ($O(\alpha^2)$ ME with exponentiation at the level of spin amplitudes) and KORALZ ($O(\alpha^2)$ ME with exponentiation) in the $Z^0 \to \mu^+\mu^-(\gamma)$ channel, $E_{test} = 1.0$ GeV, *test1* used for analysis. The plot presents the distribution of invariant mass of $\mu^+\mu^-$ pair coming from KKMC (in red, or darker-grey) and KORALZ (in green, or lighter-grey); the black line is the ratio of the two normalised distributions. The effects due to different types of exponentiation are small, albeit noticeable, and constitute 0.066% overall difference.

GENERATOR	Branching ratio		Max SDP	
n photons $\to$	0	1	0	1
KKMC	**0.83918**	**0.16082**		
KORALZ	0.83984	0.16016	0	0.0021



(as usual, complete results, with MC-TESTER booklets are available from the web page [168]).

We tested PHOTOS running with various options for photon multiplicity: fixed (two to four) order and the exponentiated versions against the reference KKMC generator.

We observed that adding subsequent iterations of photon-emission kernel improves the agreement between PHOTOS and KKMC. The difference in results between quartic-iteration and exponentiated version of PHOTOS are negligible; however, the exponentiated version is technically superior, because it may work with much lower value of the infrared cut-off parameter.

It is striking that the agreement between the exponentiated versions of PHOTOS and KKMC is best if the full $O(\alpha^2)$ exponentiated matrix element is used in KKMC; if KORALZ with exponentiated $O(\alpha)$ is used or the matrix element in KKMC is downgraded to exponentiated $O(\alpha)$ only (see the last row in the table), the agreement is not as perfect, but it is still good enough for any application we can imagine at present.

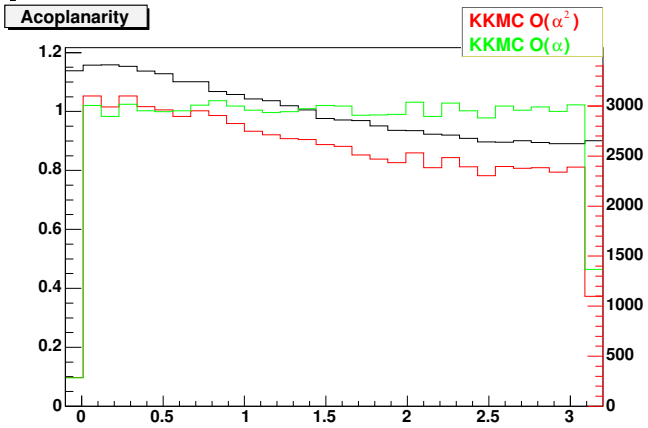
The dominant differences are visible in the branching fractions for channels with distinct numbers of photons with energies above $E_{\rm test}$. The pattern of differences may indicate that leading terms summed by an iterated PHOTOS kernel give better precision than any first-order exponentiated ME generator for this process.

To investigate this hypothesis further, we have turned our attention to acoplanarity distributions obtained from PHOTOS and KKMC. One of the effects brought by NNLO terms of $O(\alpha^2)$ ME is the asymmetry in the acoplanarity distributions. To define acoplanarity, the two planes are spanned on momenta vectors of μ^- and two hardest photons. In Fig. 18.4 we show acoplanarity distributions from the

Table 18.1: Summary of multiple-photon comparisons in $Z^0 \to \mu^+\mu^- n(\gamma)$ channel (Z^0 at resonance peak, CMS rest-frame); $E_{\rm test} = 1.0$ GeV. KKMC was used as a reference generator. For KKMC–KORALZ comparison the overall difference is 0.00066, for KKMC–PHOTOS EXP it is 0.00081, for KKMC–KKMC $O(1)_{\rm EXP}$ it is 0.00137 (all differences are for *test1*)

GENERATOR	Branching ratio					Max SDP				
	test1		test2			test1		test2		
n photons $\to$	0	1	0	1	2	0	1	0	1	2
KKMC	**.83918**	**.16082**	**.83918**	**.14816**	**.01266**					
KORAL Z	.83984	.16016	.83984	.14771	.01244	0	.0021	0	.0012	.0012
PHOTOS O(2)	.83925	.16075	.83925	.14630	.01445	0	.0067	0	.0035	.0122
PHOTOS O(3)	.83832	.16168	.83832	.14889	.01280	0	.0038	0	.0025	.0080
PHOTOS O(4)	.83836	.16164	.83836	.14871	.01293	0	.0040	0	.0027	.0058
PHOTOS EXP	.83837	.16163	.83837	.14868	.01295	0	.0041	0	.0023	.0092
KKMC $O(1)_{\rm EXP}$			.83781	.14881	.01338			0	.0099	.0467

Figure 18.4: Acoplanarity distributions (in the $Z^0 \to \mu^-\mu^+\gamma\gamma$ channel) produced by the KKMC generator running in exponentiated $O(\alpha^2)$ mode (in red, or darker-grey) and exponentiated $O(\alpha)$ mode (in green, or lighter grey); the ratio of the two distributions plotted in black. For more details see subsection 18.2.2.

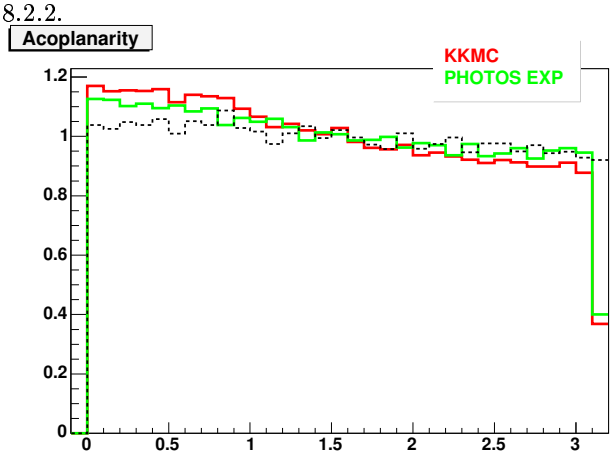


KKMC generator with $O(\alpha^2)$ and $O(\alpha)$ exponentiated matrix-element modes[5]. As expected, the distribution is flat for the $O(\alpha)$ exponentiated mode and asymmetry appears for the $O(\alpha^2)$ exponentiated mode only.

In Fig. 18.5 we present the acoplanarity distributions from the exponentiated $O(\alpha^2)$ ME mode of KKMC and exponentiated version of PHOTOS. Surprisingly, PHOTOS seems to reproduce the bulk of this NNLO effect! This rather subtle effect requires investigation. PHOTOS does not use any kind of phase-space ordering, but, at the time when the second photon is generated, the momentum of the previous one is changed and so the correlations between the directions of two photons appear. I shall not elaborate further on this effect here; however, we believe that further enhancements of precision of the PHOTOS algorithm in this aspect may be possible, for instance by introducing an asymmetry in the generation of a photon polar angle (uniform distribution is currently used in emission kernels).

[5]In order to make the effect visible, we used the following cuts: only the events with both photons in the same hemisphere as μ^- and having $p_t > 1.5$ GeV were recorded in the histogram.

Figure 18.5: Acoplanarity distributions (in the $Z^0 \to \mu^-\mu^+\gamma\gamma$ channel) produced by the KKMC generator running in exponentiated $O(\alpha^2)$ mode (in red, or darker-grey) and the exponentiated PHOTOS algorithm (in green, or lighter grey); the ratio of the two distributions is plotted as dotted black curve. For more details see subsection 18.2.2.



Let us now present the results of comparisons of exponentiated mode of PHOTOS and full $O(\alpha)$ exponentiated ME predictions of WINHAC for $W \to l\bar{\nu}_l(\gamma)$ channels. Both programs were running in exponentiated mode. The complete summary of results is presented in Table 18.2.

The details of one of these comparisons are presented in Fig. 18.6.

The overall agreement is better than per mil. The maximum value of SDP in configurations with two hard photons are larger for some tests (reaching the level of 0.0255 in the last row of the table); however, this is a channel with rather small branching fraction.

Finally, let us turn to $\tau^- \to \mu^-\nu_\tau\bar{\nu}_\mu(\gamma)$ decay. The comparison of predictions from the exponentiated version of PHOTOS and full $O(\alpha)$ ME of TAUOLA are presented in Fig. 18.7. The overall agreement is better than 0.02%.

It is interesting to compare the results presented in Fig. 18.7 with the results in Fig. 18.2, where exact TAUOLA is compared with single-photon mode of PHOTOS. One can notice that the difference between the single-photon emission mode in PHOTOS and TAUOLA is similar to the difference between exponentiated mode of PHOTOS and TAUOLA. This indicates that the dominant source of differences is due to non-leading $O(\alpha)$ terms that are missing in PHOTOS (and present in full $O(\alpha)$ ME of TAUOLA). Predictions of the single-photon mode of TAUOLA are more precise than PHOTOS predictions, even with exponentiation. Nevertheless, it must be realised that as TAUOLA does not implement exponentiation, it is limited to the generation of (at most) single hard photon emission. For cases where the topologies of configurations with two or more photons are important, the exponentiated algorithm implemented by PHOTOS will be more appropriate.

18.2.3 Tests of PHOTOS at very high energies

Because of numerical instabilities, the use of PHOTOS at LHC energies of 14 TeV was technically limited before, by a restriction on minimal $\frac{E_\gamma^{\min}}{M}$ ratio (M being the mass of the decaying particle, $E_\gamma^{\min}$ the minimal energy of generated photons). The phase space for generation had to be significantly reduced and photons of even moderate energies could not be generated.

Table 18.2: Summary of comparisons of the exponentiated algorithm of PHOTOS and WINHAC $O(\alpha)$ exponentiated for leptonic W decays. For more details, see the text.

GENERATOR	Branching ratio					Max SDP				
	test1		test2			test1		test2		
n photons $\rightarrow$	0	1	0	1	2	0	1	0	1	2
$W^+ \rightarrow \mu^+\nu_\mu(\gamma)$ at W mass, $E_{\rm test} = 1.0$ GeV										
WINHAC EXP	**.92771**	**.07229**	**.92771**	**.07007**	**.00222**					
PHOTOS EXP	.92748	.07252	.92748	.07016	.00236	0	.0029	0	.0025	.0023
$W^+ \rightarrow \mu^+\nu_\mu(\gamma)$ at W mass, $E_{\rm test} = 5.0$ GeV										
WINHAC EXP	**.96491**	**.03509**	**.96491**	**.03473**	**.00036**					
PHOTOS EXP	.96470	.03530	.96470	.04488	.00042	0	.0060	0	.006	.047
$W^+ \rightarrow e^+\nu_e(\gamma)$ at W mass, $E_{\rm test} = 1.0$ GeV										
WINHAC EXP	**.86196**	**.13804**	**.86196**	**.12943**	**.00862**					
PHOTOS EXP	.86205	.13795	.86205	.12909	.00886	0	.0022	0	.0019	.0094
$W^+ \rightarrow e^+\nu_e(\gamma)$ at W mass, $E_{\rm test} = 5.0$ GeV										
WINHAC EXP	**.93083**	**.06917**	**.93083**	**.06763**	**.00153**					
PHOTOS EXP	.93087	.06913	.93087	.06747	.00166	0	.0046	0	.0041	.0255

Figure 18.6: Multiple-photon comparison of WINHAC and PHOTOS (both running in the exponentiated mode) in the $W^+ \rightarrow \mu^+\nu_\mu n(\gamma)$ channel, $E_{\rm test} = 1.0$ GeV, analysed using *test1*. The plot (of largest SDP) presents the distribution of invariant mass of the $\mu^+\gamma$ pair coming from WINHAC (in red, or darker-grey) and PHOTOS (in green, or lighter-grey); the red and green lines practically overlap. The black line is the ratio of the two normalised distributions.

GENERATOR	Branching ratio		Max SDP	
n photons $\rightarrow$	0	1	0	1
WINHAC (EXP)	**0.92771**	**0.07229**		
PHOTOS (EXP)	0.92748	0.07252	0	0.0029

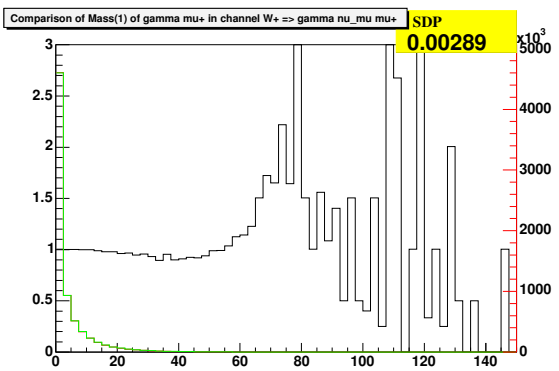
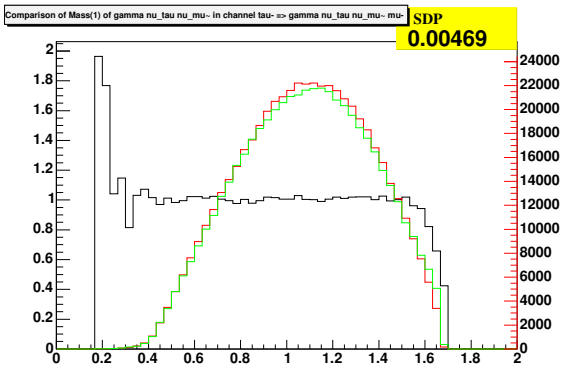


Figure 18.7: Comparison of full $O(\alpha)$ in TAUOLA and exponentiated version of PHOTOS for $\tau^- \to \mu^- \nu_\tau \bar{\nu}_\mu(\gamma)$ channel, $E_{\rm test} = 0.05$ GeV, *test1* is used. The plot below presents the distribution of invariant mass of $\nu_\tau \bar{\nu}_\mu \gamma$ from TAUOLA (in red, or darker-grey) and PHOTOS (in green, or lighter-grey); the black curve is the ratio of the two normalised distributions.

GENERATOR	Branching ratio		Max SDP	
n photons $\to$	0	1	0	1
TAUOLA	**0.98916**	**0.01084**		
PHOTOS EXP	0.98935	0.01065	0	0.0047



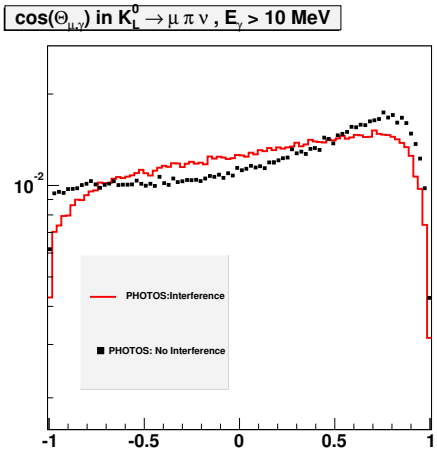
We verified that the exponentiated version of the algorithm, armed with the new kinematics-correction routine, allows it to be used to generate bremsstrahlung in decays of very energetic particles at the energy scales that indeed require very low cut on the minimal photon energy.

We performed studies using KKMC as a "host" generator for PHOTOS to produce leptonic Z-decays where the mass of the produced Z/γ intermediate state was of order 2 TeV. The value of the infrared cut-off parameter $\frac{E_\gamma^{\rm min}}{M}$ was lowered down to the value of $\sim 10^{-7}$, which allowed the generation of relatively soft photons, and high-statistics runs were completed without encountering numerical stability problems. Again, we have found excellent agreement between the exponentiated version of PHOTOS and the exponentiated $O(\alpha^2)$ KKMC. As usual, complete numerical results can be found on the web page [168].

18.2.4 Semileptonic K decays

In January 2005 our attention was drawn to a use of PHOTOS in an estimation of the effects of QED radiative corrections in high-precision measurements of K, B and D meson decays. The systematic error of results given by PHOTOS is important for measurements of elements of Cabibbo-Kobayashi-Maskawa matrix of quark mixing, because QED corrections affect the acceptance. The insufficient precision of PHOTOS observed in ref. [69] was of no surprise: for the case of semileptonic K decays, PHOTOS was prepared for use in the "crude" mode only, suitable for full phase space coverage of bremsstrahlung for detector studies, but not for shape estimations. The most important missing terms in calculations performed by PHOTOS in this case were those from QED interference. The implementation of the universal interference weight, as described in Section 13.5, allowed to improve the results significantly! The impact of the universal interference weight on the photon angle distributions in $K \to \mu\nu\pi(\gamma)$ process is presented in Fig. 18.8. It

Figure 18.8: Impact of the universal interference weight in PHOTOS on predictions for the angular distribution in $K \to \mu\pi\nu(\gamma)$ events.



seems that the majority of discrepancies between PHOTOS and the first-order ME generator KLOR have been removed in this way[6]. The complete study will, however, be documented elsewhere [216].

18.3 TAUOLA

The study similar to the one described in the previous chapter was also conducted for TAUOLA and the TAUOLA Universal Interface. The approach and methodology was similar, and the results were obtained in a similar way. For TAUOLA they are documented in [3], and for the TAUOLA Universal Interface they are in [2], therefore, I will not repeat them here.

[6]For the complete comparison, see Fig. 8b of [69] and our comparison web page [168].

Part VII

Summary and Conclusions

Summary and conclusions

This thesis has presented an overview and discussion of various aspects of computer simulations for high-energy physics, in the context of motivations, constraints and requirements of theoretical and experimental physics. In the discussion I tried to develop a consistent global view, and focus on a few, interesting from my point of view, aspects, which expose the complexity and relations of presented problematics. For some of them only a general presentation was provided, to expose their role and implications in the global view; simplifications (motivated by the nature of physics problems, and also by complexity of discussed topic) introduced in these descriptions were stressed. In the discussions of the topics that were presented in more detail I relied heavily on my personal experience; thanks to the broad spectrum of this experience, I could attempt to prepare the expressed picture.

As it was in the past, this picture may change in the future again. I hope, however, that the view presented here could be useful for some readers as a step for developing their own, possibly deeper, picture. The contexts of phenomenology of the LHC experimental collaborations (mainly: ATLAS) were predominantly present throughout this thesis, yet references to others: past, present (taking/analyzing their data now) and future experiments, such as CLEO, ALEPH, Belle, BaBar, NA49, Tevatron and Linear Collider, were also presented to enrich the global view (my personal involvement, even though indirect relations with these collaborations were used).

In the first part of the thesis the place of computer simulations in contemporary scientific methodology (and philosophy of science) was firstly established, followed by a sketch of its history (focused on the use in nuclear and high-energy physics); the potential of the simulations and the area of their use (including their important role in experimental results analysis and physics interpretation) and current trends (the use of component-based approach in programming) were also discussed. The material presented in the part *I* may, in my view, be useful as a source of general overview information (not pretending to be the complete one), and provide essential contexts and concepts, specific to high-energy physics applications, to the readers who begin their research in this area. It is the effect of my own studies and experience gathered over the last decade, while working with, and learning from, physicists in a variety of research groups, at various stages of experiment's life, and with theoretical physicists at the same time.

The issues related to the event record, the main communication channel between modules (components) of large simulation chains were discussed in the part *III*. The lack of generally-agreed, suitable and flexible event record standard (in C++) is at the source of problems that need to be faced by the authors of simulation modules, and the developers of the collaboration software chains nowadays. The most important, historical and current, proposals were briefly presented; the only widely adopted standard, `HEPEVT`, was discussed in length. The deficiencies of the existing solutions, adopted by individual developers or collaborations, were looked at - they typically originate from lack of flexibility (which was identified as the strongest requirement for the event record), and lack of communication and understanding

between groups developing the simulations (i.e. the authors of event-generators, detector simulation and physics analysis). An initial list of physics-motivated requirements for the data content and structure of the event record were formed. The requirements, the list of which is not considered as being complete, and the arguments gathered in this part of the thesis might be used as initial input for a serious discussion of the future event record standard, where the needs of all simulation groups, and physics-motivated arguments would drive the technical design of the new standard (and not the opposite). The experience gathered in the design and implementation of the `HEPEventLib` "compatibility layer" for existing event records, used successfully in `MC-TESTER`, would also be of value in the discussions of more technical details of such standardisation.

From practical, and technical, point of view, my work brought two software packages, `MC-TESTER` and `TAUOLA-PHOTOS-F`, related to three physics simulation projects: `TAUOLA`, `PHOTOS` and `TAUOLA Universal Interface`. On the experimental side, the `at2sim` simulation program was developed, parameterised, validated, then used (with my participation) to increase the confidence in the design of the ATLAS Dataflow System, an important milestone in the construction of this system, documented in the TDR document[70]. The more detailed discussion of the `at2sim` model, and its documentation, forms the content of the part *II* of my thesis.

The `TAUOLA-PHOTOS-F` system, which allow to create the required version of `PHOTOS` and `TAUOLA` packages from their master versions, maintains the rich physics content of these generators structured in a relatively compact form without code duplications etc. Its creation was the first step towards future attempts to develop packages without loss of their present physics contents; the project is based on the experience and arrangement developed in CLEO [179]. The question of the language translation for the fixed program version turns out to be relatively easy (experience has already been collected and summarised in [56]). On the contrary, the question of project continuity into further upgrades, motivated by the physics, needs to be thought over carefully. Matching the modern styles of software management and design with the strategies for further development and tests of numerical correctness of consecutive versions is a crucial issue, which certainly has to be addressed. Tools and facilities embodied directly in the existing FORTRAN codes survive such translation with difficulty. At a certain moment, the necessary strategy would require fluency in the FORTRAN, component-based Object-Orientated programming and physics content of the project by the same person.

`MC-TESTER` proved to be a recognised tool in the high-energy physics community, due to its robustness and ease of use. It was successfully used as a universal debugging and exploration tool for testing the correctness of installations of software packages in large simulation chains. `MC-TESTER` was the main and essential tool used in the studies of the precision of the `PHOTOS` algorithm presented in my thesis.

From theoretical-physics point of view, the achievement of this work is to demonstrate, for the first time, that the iterative algorithm of `PHOTOS` is capable of producing high-quality and high-precision physical estimations for QED radiative corrections. The numerical results produced by (the upgraded version of) the `PHOTOS` Monte Carlo algorithm indicate that in the case of leptonic W and Z decays, the precision of the simulations in complete multi-photon final states are at the level of (or better than) 0.1%. This conclusion is supported by numerous comparisons with precision Monte Carlo programs: `KORALZ`, `KKMC` and `WINHAC`, which generate the processes of Z and W decays on the basis of high-precision matrix-element calculations. Comparisons with fixed-order matrix-element calculations were essential as well. The reliability of the `KKMC` and `KORALZ` programs - from theoretical and technical sides - was well established by numerous applications at LEP time, documented in a multitude of publications on comparisons with the experimental data (in the case of Z decay), as well as other programs and calculations. The presented

results prove numerically that PHOTOS may be used with confidence to calculate bremsstrahlung in Z and W leptonic decays with very high precision. In the case of Higgs boson decays and leptonic decays of τ's, this precision tag is valid as well, even if the proof is not as solid, since it relies on the test at first order only. This conclusion cannot be extended to other decay channels, without further work involving a study of process-dependent matrix elements, such as the ones performed for the decays of K's [216, 159] (this is an on-going activity right now).

The use of computer-simulation based methodology, rather than rigorous, mathematical proof, for the multi-photon algorithm of PHOTOS seems to be sufficient as of now. The present-day application still do not justify the effort of performing complex analytic derivations. Wherever it was possible, PHOTOS correctness was verified and found to be very good; in other cases the tests were limited by availability of exact matrix elements for single-photon emission. More sophisticated theoretical studies are left for the future.

Similar studies (to the ones for PHOTOS) were also performed for TAUOLA and its universal interface. They are documented in [3] (for TAUOLA) and [2] (for the TAUOLA Universal Interface). A very important achievement of this study was however the construction of a testbed and methodology for organising tests of Monte Carlo event generators. This methodology, tested on the cases of PHOTOS and TAUOLA, is now ready to be used while porting the existing generators from FORTRAN to C++, and to develop the new ones. Such initiative, however, would only be sensible if experimental collaborations express their needs to have the code available in C++. There is an on-going discussion of this point with the LCG software group at CERN, in scope of the GENSER project [66]. The main topic of these discussion, HEP simulations as performed by experimental collaborations, is the one I presented in parts I and III of this thesis.

Outlook

The LHC is expected to produce first particle collisions in the Summer 2007; physics data-taking for the experimental collaborations will start soon after. The physics program of the ATLAS experiments (and others as well) involve high-precision measurements and studies in the domains of b-physics, or QCD. For that purpose dedicated, high-precision event-generators simulating the physics of the LHC would need to be made available to the experimental collaborations. The new generators, likely to be implemented in C++, may be developed for some purposes; for the others, the existing codes will certainly need to be adopted. For the latter case, the generators such as `TAUOLA` or `PHOTOS` their authors may be asked to provide a "native" C++ implementation, rather than an interface to FORTRAN codes. A "testbed" for such translation process, based on `MC-TESTER` and already explored in the tests of `TAUOLA` and `PHOTOS`, is ready for use. For the complex problem of translation and maintenance of dedicated physics codes for a variety of experimental collaboration, which requires contacts with theoretical physics on one hand, and experimental collaborations on the other, the experience (technical, and "sociological") was already gathered and documented in this thesis.

The new generators (or new, C++ translations of the existing generators) certainly need the issue of the event record standard be resolved to minimise the well-known problems with event data compatibility and interfacing. General agreement on the use of one of existing solutions, such as [118], would certainly encourage the decision of taking the effort of such translations already now. Otherwise, a proposal based on the material gathered in this thesis, enriched by further discussions, may be formulated, and its implementation provided at first. The definition of the standard (for the event record) should be performed as soon as possible, to enable the creation of new, modular generators, and "native" C++ implementation for the existing ones. Getting an ultimate agreement of all interested groups would probably not be possible, hence the decision would most probably need to be taken (and enforced) by a dedicated working group of sufficient authority. The actual choice of the event record (or: architecture) is a secondary problem: it is likely that, regardless of the choice made, a physics model will be postulated that could not be expressed by data stored in the proposed event record/architecture.

Recently, the studies that explore the feasibility of implementing NLO level of precision in `PHOTOS` (for the cases of leptonic Z decays and scalar QED) were started [155, 217] (they were only signalled in this thesis). Even though the increased precision for `PHOTOS` would not be exploited directly by the experimental collaborations, such studies may form a basis for future implementation of PHOTOS-like algorithm for QCD (`PHOTOS` as "pocket parton shower"[5]). The fact that the `PHOTOS` algorithm does not rely on the conformal symmetry of soft photon factors and the phase space of massless particles, which is crucial in the construction of exclusive-exponentiation algorithms, as used e.g. in `YFS2`, `BHLUMI`, `KKMC`, `WINHAC` or `KORALZ` is encouraging. A more thorough explanation of the mathematical side of the algorithm would, however, certainly be needed for that purpose.

Having in mind rapid changes in technology, and the plans for upgrade of the

LHC luminosity, the architecture of the ATLAS Data Flow system will need to be upgraded: it is likely that the PC-based architecture be upgraded to switch-based architecture. The `at2sim` model has already been used to validate both architectures; it could readily be used to perform the studies of the proposals of upgrade scenarios.

Bibliography

[1] A. Einstein, B. Podolsky, and N. Rosen, "Can quantum mechanical description of physical reality be considered complete?," *Phys. Rev.* **47** (1935) 777–780.

[2] M. Worek, *Spin correlations of the heavy flavors as a signal for Higgs bosons.* PhD thesis, Insitute of Physics, University of Silesia, Katowice, Poland, June, 2003.

[3] T. Pierzchala, *The extension of* TAUOLA *Monte Carlo library, simulating τ lepton decays.* PhD thesis, Silesian University, 2004. http://hpjmiady.ifj.edu.pl/~pierzchala.

[4] E. Mihova Marinova. PhD thesis, Atomic Physics Department, Sofia University. PhD thesis.

[5] P. Golonka and Z. Was, "PHOTOS as a pocket parton shower: Flexibility tests for the algorithm," hep-ph/0508015. Published in the proceedings of the "HERA and the LHC" Workshop, CERN-2005-014.

[6] http://www.desy.de/~heralhc/, CERN - DESY Workshop 2004/2005.

[7] J. Nuno, "An Internet Hotlist on Scientific Method." http://www.kn.pacbell.com/wired/fil/pages/listscientifju.html, Links to variety of articles.

[8] F. Wolfs, "Introduction to the Scientific Method." http://teacher.nsrl.rochester.edu/phy_labs/AppendixE/AppendixE.html, in 'Laboratory Experiments 1996 - 1997', Appendix E, Department of Physics and Astronomy, University of Rochester, Rochester NY 14627, USA (available on web).

[9] J. M. Garrido, *Practical Process Simulation Using Object-Oriented Techniques and C++.* ISBN 0-89006-655-8.

[10] "Wikipedia: the web encyclopedia." http://en.wikipedia.org.

[11] J. Giles, "Internet encyclopaedias go head to head," *Nature* **438** (2005) 900–901. http://www.nature.com/nature/journal/v438/n7070/full/438900a.html. doi:10.1038/438900a.

[12] D. P. Landau and K. Binder, *Monte Carlo Simulations is Statistical Physics.* Cambridge University Press, 2 ed.

[13] M. M. Woolfson and G. J. Pert, *Introduction to Computer Simulation.* University of Oxford.

[14] P. Golonka, "Rozwiazanie rownania Laplace'a metoda bladzenia przypadkowego"," January, 1998. not published, thesis for engineer degree (Praca Inzynierska), WFiTJ AGH.

[15] P. Golonka, "Kwantowe Monte Carlo," May, 1998. not published, presentation at "XXXV Sesja Studenckich Kol Naukowych Pionu Hutniczego AGH".

[16] S. Wolfram, *A new kind of science.* Wolfram Media, Inc, 2002.

[17] D. W. Heerman, *Podstawy symulcaji komputerowych w fizyce.* Wydawnictwa Naukowo-Techniczne, Warszawa, 2 ed., 1997. (original title: Computer Simulation Methods in Theoretical Physics, Springer-Verlag 1986, 1990.

[18] N. Metropolis, "The Beginning of the Monte Carlo Method," *Los Alamos Science, Special Issue* (1987), no. 15, 125.

[19] N. Metropolis and S. Ulam, "The Monte Carlo Method," *J. Amer. Stat. Assoc.* (1949), no. **44**, 335–341.

[20] D. Picard, "SIMULATION AND RECONSTRUCTION OF ELECTRON TRACKS IN A BUBBLE CHAMBER,". LAL-RI-72-10.

[21] A. M. Vaisfeld, I. E. Egorova, and G. Y. Lyubarsky, "CHARGED PARTICLE TRACK SIMULATION IN A BUBBLE CHAMBER,". KhFTI 74-21.

[22] R. J. Cence and V. J. Stenger, "MONTE CARLO SIMULATION OF NEUTRINO EVENTS IN THE NAL 15- foot BUBBLE CHAMBER,". UH-511-188-75.

[23] R. Brun, R. Hagelberg, M. Hansroul, and J. C. Lassalle, "GEANT: SIMULATION PROGRAM FOR PARTICLE PHYSICS EXPERIMENTS. USER GUIDE AND REFERENCE MANUAL,". CERN-DD-78-2-REV.

[24] R. L. Ford and W. R. Nelson, "THE EGS CODE SYSTEM: COMPUTER PROGRAMS FOR THE MONTE CARLO SIMULATION OF ELECTROMAGNETIC CASCADE SHOWERS (VERSION 3),". SLAC-0210. `http://www.slac.stanford.edu/pubs/slacreports/slac-r-210.html`.

[25] F. James, "FOWL - a General Monte-Carlo Phase Space Program," 1977. `http://preprints.cern.ch/cgi-bin/setlink?base=preprint&categ=cern&id=IT-ASDW505`, CERN Computer Centre Program Library, Long Writeup W505.

[26] "CERNLIB documentation." `http://wwwasdoc.web.cern.ch/wwwasdoc/`.

[27] S. Jadach, J. H. Kühn, and Z. Wąs, "TAUOLA: A Library of Monte Carlo programs to simulate decays of polarized tau leptons," *Comput. Phys. Commun.* **64** (1990) 275.

[28] S. Jadach, Z. Wąs, R. Decker, and J. H. Kü, "The tau decay library TAUOLA: Version 2.4," *Comput. Phys. Commun.* **76** (1993) 361.

[29] M. Jezabek, Z. Was, S. Jadach, and J. H. Kuhn, "The tau decay library TAUOLA, update with exact O(alpha) QED corrections in tau $\rightarrow$ mu. (e) neutrino anti-neutrino decay modes," *Comput. Phys. Commun.* **70** (1992) 69–76.

[30] F. James, "Monte-Carlo phase space,". CERN Program Library; CERN Yellow Report CERN-68-15. `http://preprints.cern.ch/cernrep/1968/1968-015/1968-015.html`.

[31] R. G. Sargent, "Verification and validation of simulation models,". Proceedings of the 1998 Winter Sumulation Conference.

[32] **ATLAS** Collaboration, E. Richter-Was, D. Froidevaux, and D. Poggioli, "ATLFAST 2.0 a fast simulation package for ATLAS,". ATLAS note ATL-PHYS-98-131.

[33] G. Corcella *et al.*, "HERWIG 6: An event generator for hadron emission reactions with interfering gluons (including supersymmetric processes)," *JHEP* **01** (2001) 010, `hep-ph/0011363`.

[34] T. Sjostrand *et al.*, "High-energy-physics event generation with PYTHIA 6.1," *Comput. Phys. Commun.* **135** (2001) 238, `hep-ph/0010017`.

[35] N. Wirth, *Algorithms + Data Structures = Programs*. Prentice-Hall, 1975.

[36] "International Linear Collider." `http://www.linearcollider.org`.

[37] "Extended Joint ECFA/DESY Study on Physics and Detectors for a Linear Electron-Positron Collider." `http://www.desy.de/conferences/ecfa-desy-lcext.html`.

[38] "Physics and Detectors for a 90 to 800 GeV Linear Collider: Third Workshop of the Extended ECFA/DESY Study Prague, 15th-18th November 2002." `http://www-hep2.fzu.cz/ecfadesy/ECFA-DESY_Praha2002.htm`.

[39] `http://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/HIGGS/Atlfast.html`, Original FORTRAN77 implementation of ATLFAST.

[40] T. Sjostrand and M. Bengtsson, "THE LUND MONTE CARLO FOR JET FRAGMENTATION AND e+ e- PHYSICS: JETSET VERSION 6.3: AN UPDATE," *Comput. Phys. Commun.* **43** (1987) 367.

[41] T. Sjostrand, "High-energy physics event generation with PYTHIA 5.7 and JETSET 7.4," *Comput. Phys. Commun.* **82** (1994) 74.

[42] T. A. Collaboration, *ATLAS Computing Technical Design Report*. CERN, July, 2005.

[43] "Geant4 toolkit." `http://cern.ch/geant4`.

[44] `http://www.hep.ucl.ac.uk/atlas/atlfast/`, The web-page of ATLFAST as ATHENA component.

[45] T. A. Collaboration, *ATLAS detector and physics performance Technical Design Report, volume II*. Tech. Design Rep. ATLAS. CERN, May, 1999.

[46] "ATLAS Geant4 Photo Album." `http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/OO/simulation/geant4/photoalbum.html`.

[47] M. A. Hofmann, "Criteria for Decomposing Systems Into Components in Modeling and Simulation: Lessons Learned with Military Simulations," *SIMULATION* **80** (2004), no. 7-8, 357–365. `http://sim.sagepub.com/cgi/content/abstract/80/7-8/357`. doi:10.1177/0037549704049876.

[48] A. F. Sisti, "LARGE-SCALE BATTLEFIELD SIMULATION USING A MULTI-LEVEL MODEL INTEGRATION METHODOLOGY,". http://www.rl.af.mil/tech/papers/docs/Large-Scale.doc.

[49] D. L. Parnas, "On the Criteria To Be Used in Decomposing Systems into Modules," *Communications of the ACM* **15** (December, 1972). http://www.cs.virginia.edu/~cs340/uva/papers/parnas.12.72.pdf.

[50] N. Oses, M. Pidd, and R. J. Brooks, "Critical issues in the development of component-based discrete simulations," *Simulation Modelling Practice and Theory* **12** (2004) 495–514. doi:10.1016/j.simpat.2004.06.005.

[51] L. software architecture group, *GAUDI, LHCb Data Processing Applications Framework, Architecture Design Document.* http://lhcb-comp.web.cern.ch/lhcb-comp/Meetings/offline/pdf/add.pdf.

[52] S. Robinson *et al.*, "Simulation model reuse: definitions, benefits and obstacles," *Simulation Modelling Practice and Theory* **12** (2004) 479–494. doi:10.1016/j.simpat.2003.11.006.

[53] M. Marino *et al.*, "Monte Carlo simulation in the nineties,". Prepared for INFN Eloisatron Project: 9th Workshop: Perspectives for New Detectors in Future Supercolliders, Erice, Italy, 17-24 Oct 1989.

[54] F. Anselmo *et al.*, "COSMOS: A Comprehensive super Monte Carlo system,". In *Erice 1990, Proceedings, Data structures for particle physics eperiments* 70-76.

[55] S. Jadach, B. F. L. Ward, and Z. Was, "The precision Monte Carlo event generator KK for two- fermion final states in e+ e- collisions," *Comput. Phys. Commun.* **130** (2000) 260, hep-ph/9912214.

[56] P. Golonka, "PHOTOS+ - a C++ Implementation of a Universal Monte Carlo Algorithm for QED Radiative Corrections in Particle's Decays," Master's thesis, Faculty of Nuclear Physics and Techniques, AGH University of Science and Technology, June, 1999. Written under the supervision of Z. Was. http://cern.ch/Piotr.Golonka/MC/photos.

[57] "ATHENA: ATLAS Offline Computing framework." http://atlas-computing.web.cern.ch/atlas-computing/packages/athenaCore.php.

[58] "Qt: a comprehensive C++ application development framework." http://www.trolltech.com/products/qt.

[59] "GAUDI project." http://cern.ch/proj-gaudi.

[60] "ROOT: an object-oriented data analysis framework." http://root.cern.ch.

[61] *Athena Developer Guide (8.0.0) DRAFT.* http://atlas-computing.web.cern.ch/atlas-computing/packages/athenaCore.php.

[62] "AthenaFramework on Atlas Wiki." https://uimon.cern.ch/twiki/bin/view/Atlas/AthenaFramework.

[63] "LHCb: Software frameworks." http://lhcb-comp.web.cern.ch/lhcb-comp/Frameworks/.

[64] "ALICE Offline software, AliRoot framework." http://aliceinfo.cern.ch/Offline.

[65] "EDM: Event Data Model framework for CMS off-line computing." `https://uimon.cern.ch/twiki/bin/view/CMS/EDM`.

[66] "GENSER: LCG Generator Services Subproject." `http://lcgapp.cern.ch/project/simu/generator/`.

[67] "SHERPA:Simulation of High Energy Reactions of PArticles." `http://www.sherpa-mc.de`, simulation package for physics collider experiments: LEP, HERA, LHC, Tevatron, TESLA.

[68] I. Zacharov, "Towards the data abstraction in HEP,". In *Erice 1990, Proceedings, Data structures for particle physics experiments* 141-146.

[69] T. C. Andre, "Radiative corrections to K0(l3) decays," `hep-ph/0406006`.

[70] T. A. Collaboration, *ATLAS High-Level Trigger, Data Acquisition and Controls Technical Design Report.* CERN, June, 2003.

[71] T. A. Collaboration, *ATLAS High-Level Triggers, DAQ and DCS : Technical Proposal.* CERN, 2000. `http://atlas.web.cern.ch/Atlas/GROUPS/DAQTRIG/SG/TP/tp_doc.html`.

[72] B. P. Kersevan and E. Richter-Was, "Processing generated events with TAUOLA and PHOTOS using the Athena interface." `http://phyweb.lbl.gov/~ianh/monte/Generators/Photos/index.html`.

[73] "ATLAS Level 2 pilot project." `http://atlasinfo.cern.ch/Atlas/GROUPS/DAQTRIG/L2PILOT/l2pilot.html`.

[74] `http://ptolemy.eecs.berkeley.edu`, Ptolemy project.

[75] R. Hauser, "The Atlas Level 2 Reference Software,". ATL-DAQ-2000-019.

[76] M. Dobson *et al.*, "Ptolemy simulation of the ATLAS level-2 trigger,". ATL-DAQ-2000-039.

[77] R. W. Dobinson *et al.*, "Testing and Modeling Ethernet Switches and Networks for Use in ATLAS High-level Triggers,". CERN-OPEN-2000-310 ; In: Workshop on Network-Based Data Acquisition and Event-Building DAQ 2000, Lyon, France, 20 Oct 2000.

[78] R. Dobinson, K. Korcyl, and F. Saka, "Modeling large Ethernet networks using parametrized switches.,". OPNETWORK 2000 conference, Washington DC, 28 Aug - 1 Sep 2000.

[79] F. Saka, *Ethernet for the ATLAS Second Level Trigger.* PhD thesis, Royal Holloway University of London, 2001. RHCPP 01-31.

[80] P. Golonka, K. Korcyl, and F. Saka, "Modeling large Ethernet networks for the ATLAS high level trigger system using parameterized models of switches and nodes,". Prepared for 12th IEEE-NPSS Real Time Conference, Valencia, Spain, 4-8 Jun 2001.

[81] A. Morton, "zc and cyclesoak: Tools for accurately measuring system load and TCP efficiency." `http://www.zip.com.au/~akpm/linux/#zc`.

[82] "OProfile: system-wide profiler for Linux systems." `http://oprofile.sourceforge.net/about/`.

[83] P. Golonka, "Using OProfile for DataCollection performance measurements," tech. rep. ATL-DQ-EN-0013. https://edms.cern.ch/document/391591.

[84] P. Golonka, "Linux network performance study for the ATLAS Data Flow System," tech. rep., CERN. ATL-DQ-TR0001. https://edms.cern.ch/document/368844.

[85] C. Meirosu *et al.*, "Native 10 Gigabit Ethernet experiments over long distances," *Future Generation Computer Systems* **21** (April, 2004) 457–468. also in: ATL-D-ON-0001, ATL-D-TN-0001. doi:10.1016/j.future.2004.10.003.

[86] "Interrupt Moderation Using Intel Gigabit Ethernet Controllers," tech. rep., Intel. Intel Application Note AP-450. http://www.intel.com/design/network/applnots/ap450.htm.

[87] R. Prasad, M. Jain, and C. Dovrolis, "Effects of Interrupt Coalescence on Network Measurements,". punlished in Proceedings of 5th Passive and Active Measurements Workshop, PAM2004, April 19-20, 2004, Antibes Juan-les-Pins, France. http://www.pam2004.org/papers/265.pdf.

[88] "at2sim: Discrete-event computer model of the ATLAS Trigger/DAQ system." http://cern.ch/at2sim, AT2SIM project web-page.

[89] P. Golonka and K. Korcyl, "Calibration of the ATLAS Data Collection component models.," tech. rep., CERN. ATL-DQ-TP-0001. https://edms.cern.ch/document/391590.

[90] H. B. et al, "The base-line DataFlow system of the ATLAS Trigger and DAQ,". ATL-DAQ-2004-006.

[91] H. B. et al, "ATLAS TDAQ: A Network-based architecture,". EDMS Note ATL-DQ-EN-0014.

[92] F. Barnes, R. Beuran, R. Dobinson, M. LeVine, B. Martin, J. Lokier, and C. Meirosu, "Testing Ethernet Networks for the ATLAS Data Collection System," *IEEE Trans. Nucl. Sci.* **49** (April, 2002) 516–520. http://www.cs.ukc.ac.uk/pubs/2002/1385.

[93] **ATLAS** Collaboration, J. e. a. Vermenulen, "ATLAS DataFlow: the Read-Out Subsystem, Results from Trigger and Data Acquisition System Testbed Studies and from Modeling,". ATL-DAQ-CONF-2005-025, Proceedings of the 14th IEEE NPSS Real Time Conference, RT2005, Stockholm, Sweden; Submitted to Trans. Nucl. Sci.

[94] **ATLAS** Collaboration, G. e. a. Unel, "Performance of the ATLAS DAQ Dataflow System,". CERN-2005-002-V-1, Proceedings of the CHEP 2004 Conference, Interlaken, Switzerland.

[95] K. Desch, A. Imhof, Z. Was, and M. Worek, "Probing the CP nature of the Higgs boson at linear colliders with tau spin correlations: The case of mixed scalar-pseudoscalar couplings," *Phys. Lett.* **B579** (2004) 157–164, hep-ph/0307331.

[96] T. Pierzchała, E. Richter-Wąs, Z. Wąs, and M. Worek, "Spin effects in tau-lepton pair production at LHC," *Acta Phys. Polon.* **B32** (2001) 1277, hep-ph/0101311.

[97] Z. Was, "Spin polarization and the Einstein-Podolsky-Rosen paradox in the Monte Carlo event records," hep-ph/0203217.

[98] "A turnkey service to secure communications, based on Quantum Cryptography, June 2005." `http://www.idquantique.com`, Quantique S.A, Switzerland.

[99] "MagiQ QPN(tm) Security Gateway Quantum Key Distribution (QKD) System." `http://www.magiqtech.com`, MagiQ Tech, USA.

[100] "Quantum Physics to the Rescue." `http://www.csoonline.com/read/050105/machine.html`, CSO Magazine, June 2005.

[101] S. Jadach, B. F. L. Ward, and Z. Was, "Global positioning of spin GPS scheme for half-spin massive spinors," *Eur. Phys. J.* **C22** (2001) 423–430, `hep-ph/9905452`.

[102] S. Jadach, B. F. L. Ward, and Z. Was, "Coherent exclusive exponentiation CEEX: The case of the resonant e+ e- collision," *Phys. Lett.* **B449** (1999) 97–108, `hep-ph/9905453`.

[103] J. C. Collins, "SPIN CORRELATIONS IN MONTE CARLO EVENT GENERATORS," *Nucl. Phys.* **B304** (1988) 794.

[104] P. Richardson, "Spin correlations in Monte Carlo simulations," *JHEP* **11** (2001) 029, `hep-ph/0110108`.

[105] M. A. Dobbs *et al.*, "Les Houches guidebook to Monte Carlo generators for hadron collider physics," `hep-ph/0403045`.

[106] T. Sjostrand *et al.*, "Z physics at LEP1, chapter *Generators*." CERN 89-08, vol 3, p.327, September, 1989. `http://cdsweb.cern.ch/search.py?recid=367653&ln=en`.

[107] "LEP 2 QCD Event Generators Working Group home page." `http://ppewww.ph.gla.ac.uk/lep2/`.

[108] I. G. Knowles *et al.*, "QCD Event Generators," `hep-ph/9601212`.

[109] P. Golonka *et al.*, "The tauola-photos-F environment for the TAUOLA and PHOTOS packages, release II," *Comput. Phys. Commun.* **174** (May, 2006) 818–835, `hep-ph/0312240`.

[110] G. P. Yost *et al.*, "REVIEW OF PARTICLE PROPERTIES: PARTICLE DATA GROUP," *Phys. Lett.* **B204** (1988) 1–486.

[111] **Particle Data Group** Collaboration, L. Montanet *et al.*, "Review of particle properties. Particle Data Group," *Phys. Rev.* **D50** (1994) 1173–1823.

[112] **Particle Data Group** Collaboration, C. Caso *et al.*, "Review of particle physics," *Eur. Phys. J.* **C3** (1998) 1.

[113] **Particle Data Group** Collaboration, S. Eidelman *et al.*, "Review of particle physics," *Phys. Lett.* **B592** (2004) 242.

[114] `http://pdg.lbl.gov`, An up-to-date version of the PDG.

[115] L. Garren, "StdHep 5.03 Monte Carlo Standardization at FNAL Fortran and C Implementation," June, 2004. `http://cepa.fnal.gov/psm/stdhep`.

[116] E. Boos *et al.*, "Generic user process interface for event generators," `hep-ph/0109068`.

[117] H. Plothow-Besch, "PDFLIB: A Library of all available parton density functions of the nucleaon, the pion, and the photon and the corresponding alpha-s calculation," *Comput. Phys. Commun.* **75** (1993) 396–416. http://consult.cern.ch/writeup/pdflib.

[118] M. Dobbs and J. B. Hansen, "The HepMC C++ Monte Carlo event record for High Energy Physics," *Comput. Phys. Commun.* **134** (2001) 41–46.

[119] F. Anselmo *et al.*, "MEGA: Monte Carlo Event Generator Adaptor," *Part. World* **2** (1991) 108–116.

[120] O. Dir Rosa, B. Van Eijk, F. Carminati, I. Zacharov, and D. Hatzifotiadou, "Information modeling for Monte Carlo event generators: Standardization within a universal framework for event generators an detector simulation,". In *Erice 1990, Proceedings, Data structures for particle physics eperiments* 83-90.

[121] S. Protopopescu, "MC++ Interface." http://www-d0.fnal.gov/~serban/mcpp/.

[122] "StdHep in C++." http://www-cpd.fnal.gov/psm/stdhep/c++/.

[123] "Dag - Directed Acyclic graph - A Shortened Discussion." http://www-cpd.fnal.gov/psm/stdhep/c++/dag.txt.

[124] "CLHEP: A Class Library for High Energy Physics." http://cern.ch/clhep.

[125] "HepMC: a C++ Event Record for Monte Carlo Generators." http://cern.ch/hepMC.

[126] "ThePEG: Toolkit for High Energy Physics Event Generation." http://www.thep.lu.se/ThePEG/.

[127] "The Herwig++ Event Generator." http://hepforge.cedar.ac.uk/herwig/.

[128] M. Bertini, L. Lonnblad, and T. Sjostrand, "Pythia version 7-0.0: A proof-of-concept version," *Comput. Phys. Commun.* **134** (2001) 365–391, hep-ph/0006152.

[129] "ThePEG class hierarchy (reference doxygen-generated documentation)." http://www.thep.lu.se/ThePEG/Doc/refman-html/hierarchy.html.

[130] T. Gleisberg *et al.*, "SHERPA 1.alpha, a proof-of-concept version," *JHEP* **02** (2004) 056, hep-ph/0311263.

[131] M. A. Thomson, "The influence of the experimental methodology on the QED theoretical uncertainties on the measurement of M(W) at LEP," *JHEP* **08** (2003) 009, hep-ph/0307043.

[132] M. L. Mangano, M. Moretti, F. Piccinini, R. Pittau, and A. D. Polosa, "ALPGEN, a generator for hard multiparton processes in hadronic collisions," *JHEP* **07** (2003) 001, hep-ph/0206293.

[133] B. P. Kersevan and E. Richter-Wąs, "The Monte Carlo event generator AcerMC version 1.4 with interfaces to PYTHIA 6.2 and HERWIG 6.3," *Comput. Phys. Commun.* **149** (2003) 142. http://cern.ch/Borut.Kersevan/AcerMC.Welcome.html.

[134] P. J. Stephens, *Computer simulations of High Energy Physics*. PhD thesis, Trinity College, University of Cambridge, May, 2004. hep-ph/0408363.

[135] F. Anselmo *et al.*, "Event generators for LHC,".

[136] B. Andersson, G. Gustafson, G. Ingelman, and T. Sjostrand, "PARTON FRAGMENTATION AND STRING DYNAMICS," *Phys. Rept.* **97** (1983) 31.

[137] G. Marchesini and B. R. Webber, "MONTE CARLO SIMULATION OF GENERAL HARD PROCESSES WITH COHERENT QCD RADIATION," *Nucl. Phys.* **B310** (1988) 461.

[138] J.-C. Winter, F. Krauss, and G. Soff, "A modified cluster-hadronization model," *Eur. Phys. J.* **C36** (2004) 381–395, hep-ph/0311085.

[139] "QQ - The CLEO Event Generator." http://www.lns.cornell.edu/public/CLEO/soft/QQ.

[140] M. Dobbs *et al.*, "The QCD/SM working group: Summary report," hep-ph/0403100.

[141] A. M. Moraes, "Minimum bias interactions and the underlying event." http://www.shef.ac.uk/physics/research/pppa/research/atlas/ukphys/meeting8/slides/moraes

[142] J. Butterworth, "Modelling Underlying Events and Minimum Bias Events." http://www.wlap.org/file-archive/atlas/20040622-umwlap002-06-butterworth.pdf.

[143] E. Barberio, B. van Eijk, and Z. Was, "PHOTOS: A Universal Monte Carlo for QED radiative corrections in decays," *Comput. Phys. Commun.* **66** (1991) 115.

[144] E. Barberio and Z. Was, "PHOTOS: A Universal Monte Carlo for QED radiative corrections. Version 2.0," *Comput. Phys. Commun.* **79** (1994) 291–308.

[145] **CDF** Collaboration, V. M. Abazov *et al.*, "Combination of CDF and D0 results on W boson mass and width," *Phys. Rev.* **D70** (2004) 092008, hep-ex/0311039.

[146] **OPAL** Collaboration, G. Abbiendi *et al.*, "A study of W+ W- gamma events at LEP," *Phys. Lett.* **B580** (2004) 17–36, hep-ex/0309013.

[147] **DELPHI** Collaboration, J. Abdallah *et al.*, "Measurement of the e+ e- –> W+ W- gamma cross-section and limits on anomalous quartic gauge couplings with DELPHI," *Eur. Phys. J.* **C31** (2003) 139–147, hep-ex/0311004.

[148] **NA48** Collaboration, A. Lai *et al.*, "Measurement of the branching ratio of the decay K(L) –> pi+- e-+ nu and extraction of the CKM parameter |V(us)|," *Phys. Lett.* **B602** (2004) 41–51, hep-ex/0410059.

[149] **KTeV** Collaboration, T. Alexopoulos *et al.*, "Measurements of the branching fractions and decay distributions for K(L) –> pi+- mu-+ nu gamma and K(L) –> pi+- e-+ nu gamma," *Phys. Rev.* **D71** (2005) 012001, hep-ex/0410070.

[150] **Belle** Collaboration, A. Limosani *et al.*, "Measurement of inclusive charmless semileptonic B-meson decays at the endpoint of the electron momentum spectrum," hep-ex/0504046.

[151] **BABAR** Collaboration, B. Aubert *et al.*, "Measurements of moments of the hadronic mass distribution in semileptonic B decays," *Phys. Rev.* **D69** (2004) 111103, hep-ex/0403031.

[152] **FOCUS** Collaboration, J. M. Link *et al.*, "Measurement of the doubly Cabibbo suppressed decay D0 -> K+ pi- and a search for charm mixing," hep-ex/0412034.

[153] P. Golonka and Z. Was, "PHOTOS Monte Carlo: A precision tool for QED corrections in Z and W decays," *Eur. Phys. J.* **C45** (2006) 97–107, hep-ph/0506026. doi:10.1140/epjc/s2005-02396-4.

[154] P. Golonka, E. Richter-Was, and Z. Was, "The tauola-photos-F environment for versioning the TAUOLA and PHOTOS packages," hep-ph/0009302.

[155] P. Golonka and Z. Was, "Next to Leading Logarithms and PHOTOS Monte Carlo.,". in preparation.

[156] Z. Was, "Radiative corrections,". CERN-TH-7154-94; Written on the basis of lectures given at the 1993 European School of High Energy Physics, Zakopane, Poland, 12-25 Sep 1993.

[157] E. Richter-Was, "Hard photon bremsstrahlung in the process p p → Z0 / gamma* → lepton+ lepton-: A background for the intermediate mass Higgs," *Z. Phys.* **C64** (1994) 227–240.

[158] E. Richter-Was, "Hard Bremsstrahlung photons in the t anti-t production and decay: A background for the intermediate Higgs search," *Z. Phys.* **C61** (1994) 323–340.

[159] "Radiative corrections in B, D and K meson decays" workshop and CKM05 Conference, La Jolla, March 2005,.

[160] S. Jadach, B. F. L. Ward, and Z. Wąs, "The Monte Carlo program KORALZ version 4.0 for the lepton or quark pair production at LEP/SLC energies," *Comput. Phys. Commun.* **79** (1994) 503.

[161] P. H. Eberhard *et al.*, "THE tau POLARIZATION MEASUREMENT AT LEP,". To appear in the Proceedings of the Workshop on Z Physics at LEP, edited by G. Altarelli, R. Kleiss and V. Verzegnassi (CERN-89-08 v.1-3) held in Geneva, Switzerland, Feb 20-21 and May 8-9, 1989. Published in LEP Physics Wrkshp.1989:v.1:235-266.

[162] F. A. Berends, R. Kleiss, and S. Jadach, "RADIATIVE CORRECTIONS TO MUON PAIR AND QUARK PAIR PRODUCTION IN ELECTRON - POSITRON COLLISIONS IN THE Z(0) REGION," *Nucl. Phys.* **B202** (1982) 63.

[163] F. A. Berends, R. Kleiss, and S. Jadach, "MONTE CARLO SIMULATION OF RADIATIVE CORRECTIONS TO THE PROCESSES E+ E- → MU+ MU- AND E+ E- → ANTI-Q Q IN THE Z0 REGION," *Comput. Phys. Commun.* **29** (1983) 185–200.

[164] Z. Was, "Gauge invariance, infrared / collinear singularities and tree level matrix element for e+ e- -> nu/e anti-nu/e gamma gamma," hep-ph/0406045.

[165] S. Jadach, B. F. L. Ward, and Z. Was, "The Monte Carlo program KORALZ, version 3.8, for the lepton or quark pair production at LEP / SLC energies," *Comput. Phys. Commun.* **66** (1991) 276–292.

[166] J. J. Hernandez *et al.*, "Review of particle properties. Particle Data Group," *Phys. Lett.* **B239** (1990) 1–515.

[167] G. Nanava and Z. Was, "How to use SANC to improve the PHOTOS Monte Carlo simulation of bremsstrahlung in leptonic W-boson decays," *Acta Phys. Polon.* **B34** (2003) 4561–4570, hep-ph/0303260.

[168] P. Golonka and Z. Was, "Studies of precision of the PHOTOS algorithm." http://cern.ch/Piotr.Golonka/MC/PHOTOS-MCTESTER.

[169] D. Yennie, S. Frautschi, and H. Suura, "The infrared divergence phenomena and high-energy process," *Ann. Phys.* **13** (1961) 379–452.

[170] A. Andonov, S. Jadach, G. Nanava, and Z. Was, "Comparison of SANC with KORALZ and PHOTOS," *Acta Phys. Polon.* **B34** (2003) 2665–2672, hep-ph/0212209.

[171] S. Jadach and Z. Was, "KORALB version 2.1: An Upgrade with TAUOLA library of tau decays," *Comput. Phys. Commun.* **64** (1991) 267–274.

[172] Z. Wąs and M. Worek, "Transverse spin effects in $H/A \to \tau^+\tau^-$; $\tau^\pm \to \nu X^\pm$, Monte Carlo approach," *Acta Phys. Polon.* **B33** (2002) 1875, hep-ph/0202007.

[173] G. R. Bower, T. Pierzchała, Z. Wąs, and M. Worek, "Measuring the Higgs boson's parity using tau → rho nu," *Phys. Lett.* **B543** (2002) 227, hep-ph/0204292.

[174] K. Desch, Z. Wąs, and M. Worek, "Measuring the Higgs boson parity at a linear collider using the tau impact parameter and tau -> rho nu decay," *Eur. Phys. J.* **C29** (2003) 491, hep-ph/0302046.

[175] S. Jadach and Z. Was, "QED O (ALPHA**3) RADIATIVE CORRECTIONS TO THE REACTION E+ E- → TAU+ TAU- INCLUDING SPIN AND MASS EFFECTS. (ERRATUM)," *Acta Phys. Polon.* **B15** (1984) 1151.

[176] "CERN Workshop on Monte Carlo tools for the LHC, session on decay packages." http://agenda.cern.ch/fullAgenda.php?ida=a031540, Chair W. Pokorski, N. Brook, July 7 - Aug 1 2003, CERN Geneva,.

[177] "HERWIG: Monte Carlo package for simulating Hadron Emission Reactions With Interfering Gluons." http://hepwww.rl.ac.uk/theory/seymour/herwig/.

[178] Z. Was, "Observables with tau leptons at LHC and LC: Structure of event records and Monte Carlo algorithms," *Nucl. Instrum. Meth.* **A534** (2004) 260–264, hep-ph/0402129.

[179] **CLEO** Collaboration, "courtesy of A. Weinstein," 1997. http://www.cithep.caltech.edu/~ajw/korb_doc.html.

[180] **ALEPH** Collaboration, B. Bloch. to contact send a mail to Brigitte.Bloch-Devaux@cern.ch.

[181] P. Golonka. in preparation.

[182] "Qt Class Libraries." http://www.trolltech.com/products/qt/qtlibrary.htm.

[183] "STL: Standard Template Library." http://www.sgi.com/tech/stl/.

[184] "make." http://www.opengroup.org/onlinepubs/009695399/utilities/make.html, The Open Group Base Specifications Issue 6 IEEE Std 1003.1, 2004 Edition.

[185] "GNU automake." http://www.gnu.org/software/automake/.

[186] "GNU autoconf." http://www.gnu.org/software/autoconf/.

[187] S. Jadach and P. Sawicki, "mFOAM-1.02: A compact version of the cellular event generator FOAM," physics/0506084.

[188] "CMT: Configuration Management Tool." http://www.cmtsite.org/.

[189] B. Jacobsen, "SRT: Software Release Tools." http://www.slac.stanford.edu/BFROOT/www/Computing/Environment/Tools/SRT.

[190] "CVS: Concurrent Version System." http://www.nongnu.org/cvs/.

[191] "Subversion version control system." http://subversion.tigris.org/.

[192] "Valgrind suite of debugging and profiling tools." http://valgrind.org/.

[193] "Doxygen: documentation system for C++, C, ..." http://www.stack.nl/~dimitri/doxygen/.

[194] "The PVSS JCOP Framework Project." http://itcobe.web.cern.ch/itcobe/Projects/Framework/welcome.html.

[195] "Umbrello UML Modeller." http://uml.sourceforge.net/.

[196] "Dia: gtk+ based diagram creation program." http://www.gnome.org/projects/dia/.

[197] "The Python programming language." www.python.org.

[198] "CLEO software documentation." http://www.lns.cornell.edu/public/CLEO/soft.

[199] S. Jadach and Z. Was, "Koralb: An Upgrade to version 2.4," *Comput. Phys. Commun.* **85** (1995) 453–462.

[200] P. Golonka, T. Pierzchala, and Z. Was, "MC-TESTER: A universal tool for comparisons of Monte Carlo predictions for particle decays in high energy physics," *Comput. Phys. Commun.* **157** (2004) 39–62, hep-ph/0210252.

[201] R. Decker, M. Finkemeier, P. Heiliger, and H. H. Jonsson, "Tau decays into four pions," *Z. Phys.* **C70** (1996) 247–254, hep-ph/9410260.

[202] A. E. Bondar *et al.*, "Novosibirsk hadronic currents for tau –> 4pi channels of tau decay library TAUOLA," *Comput. Phys. Commun.* **146** (2002) 139–153, hep-ph/0201149.

[203] H. Czyz and J. Kuhn, "Four pion final states with tagged photons at electron positron colliders," *Eur.Phys.J.* **C 18** (2001) 497, hep-ph/0008262.

[204] "GCC, the GNU Compiler Collection." `http://gcc.gnu.org`, homepage.

[205] Talks by B. Kersevan and M. Worek, presented at the *'Workshop on Monte Carlo tools for the LHC'*, CERN, July 2003.

[206] G. Passarino, R. Pittau, and S. Jadach, eds., *LEP2 Monte Carlo Workshop: Report of the Working Groups on Precision Calculations for LEP2 Physics*, CERN. 2000.

[207] `http://www.desy.de/conferences/ecfa-desy-lcext.html`, Extended Joint ECFA/DESY Study on Physics and Detectors for a Linear Electron-Positron Collider.

[208] "MC-TESTER homepage." `http://cern.ch/Piotr.Golonka/MC/MC-TESTER`.

[209] `http://root.cern.ch/root/htmldoc/TH1.html#TH1:KolmogorovTest`.

[210] `http://root.cern.ch/root/htmldoc/src/TH1.cxx.html#TH1:KolmogorovTest`.

[211] S. Jadach, B. F. L. Ward, and Z. Wąs, "The Monte Carlo program KORALZ, for the lepton or quark pair production at LEP/SLC energies: From version 4.0 to version 4.04," *Comput. Phys. Commun.* **124** (2000) 233–237, `hep-ph/9905205`.

[212] W. Placzek and S. Jadach, "Multiphoton radiation in leptonic W-boson decays," *Eur. Phys. J.* **C29** (2003) 325–339, `hep-ph/0302065`.

[213] **ALEPH** Collaboration, M. Davier and C.-z. Yuan, "Measurement of branching fractions in tau decays," *eConf* **C0209101** (2002) TU06, `hep-ex/0211057`.

[214] **CLEO** Collaboration, S. Anderson *et al.*, "Hadronic structure in the decay tau- –> pi- pi0 nu/tau," *Phys. Rev.* **D61** (2000) 112002, `hep-ex/9910046`.

[215] A. Andonov *et al.*, "Project SANC (former CalcPHEP): Support of analytic and numeric calculations for experiments at colliders," `hep-ph/0209297`.

[216] P. Golonka, L. Litov, E. Mihova, and Z. Was, 2005. In preparation.

[217] "Photos at NLO." `http://cern.ch/Piotr.Golonka/MC/PHOTOS-MCTESTER/AtNLO`.