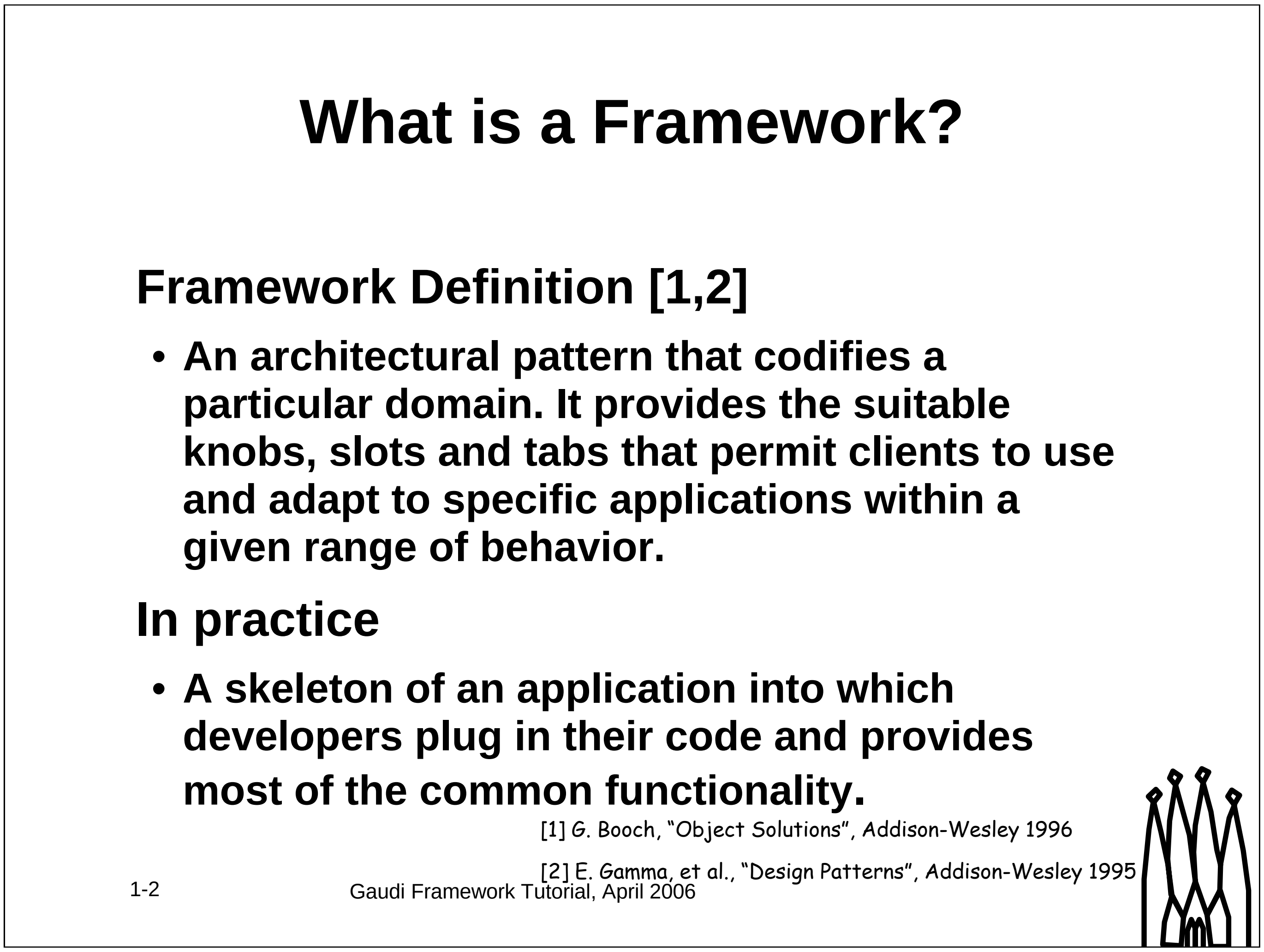
1

Introduction

Gaudi Framework Tutorial, April 2006

Schedule:

Timing	Topic
20 minutes	Lecture
0 minutes	Practice
20 minutes	Total



What is a Framework?

Framework Definition [1,2]

- An architectural pattern that codifies a particular domain. It provides the suitable knobs, slots and tabs that permit clients to use and adapt to specific applications within a given range of behavior.

In practice

- A skeleton of an application into which developers plug in their code and provides most of the common functionality.

[1] G. Booch, "Object Solutions", Addison-Wesley 1996

[2] E. Gamma, et al., "Design Patterns", Addison-Wesley 1995

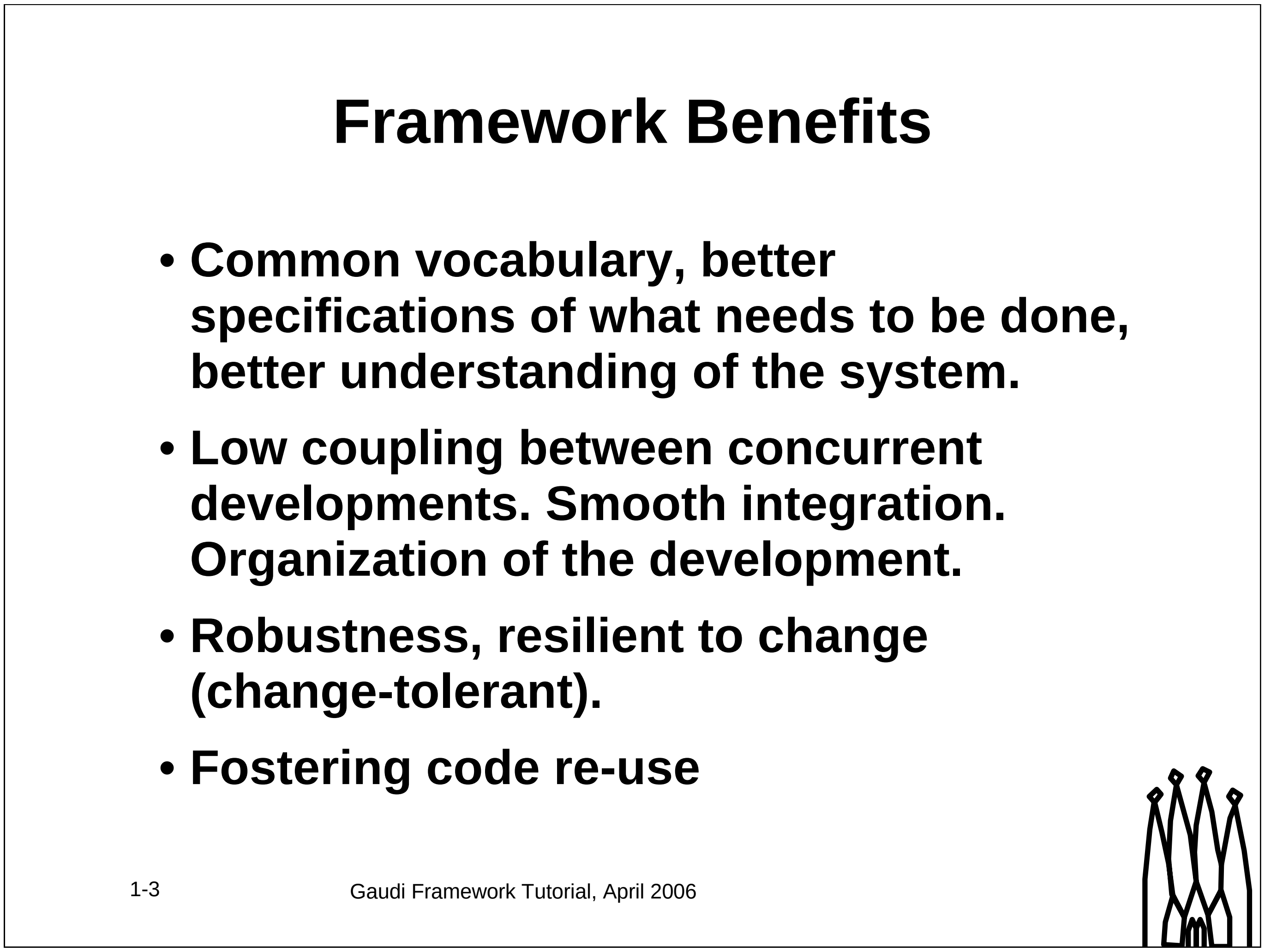
1-2 Gaudi Framework Tutorial, April 2006

What is a Framework

We have developed the Gaudi object-oriented framework with the intention of using it to develop all event data processing applications running in the various processing environments. Examples include the high level triggers that acquire their data directly from the on-line system, the event simulation software that runs off-line in a batch environment and the event visualization software which is used interactively. The basis of the framework is the architecture which defines the basic components and how they are interfaced. The framework is real software that implements the architecture and ensures that its design features are respected.

Use of the framework in all applications will help to ensure the integrity of the overall software design and will result in maximum reuse of the core software components.

Although GAUDI has been developed in the context of the LHCb experiment, it has been designed to be easily customizable, so that it can be adapted to the different tasks and integrated with components from other frameworks. It can easily be adapted for use in other experiments.



Framework Benefits

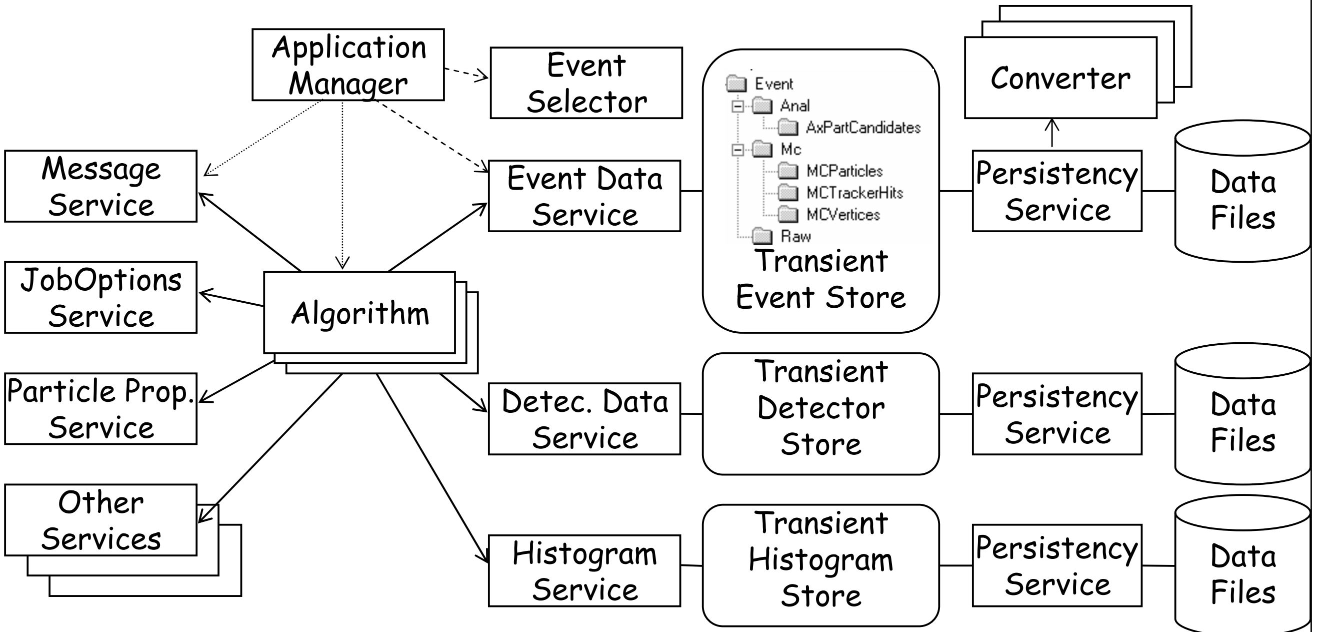
- Common vocabulary, better specifications of what needs to be done, better understanding of the system.
- Low coupling between concurrent developments. Smooth integration. Organization of the development.
- Robustness, resilient to change (change-tolerant).
- Fostering code re-use

1-3 Gaudi Framework Tutorial, April 2006

Framework benefits

The main benefit of adopting a framework is to give a frame to the developments of the different groups and people, adopt a common vocabulary, etc. Without a framework, the different contributions will be very difficult to integrate into a coherent application like the reconstruction program. It is secondary in importance which concrete framework you adopt, the important thing is to have one.

Gaudi Object Diagram

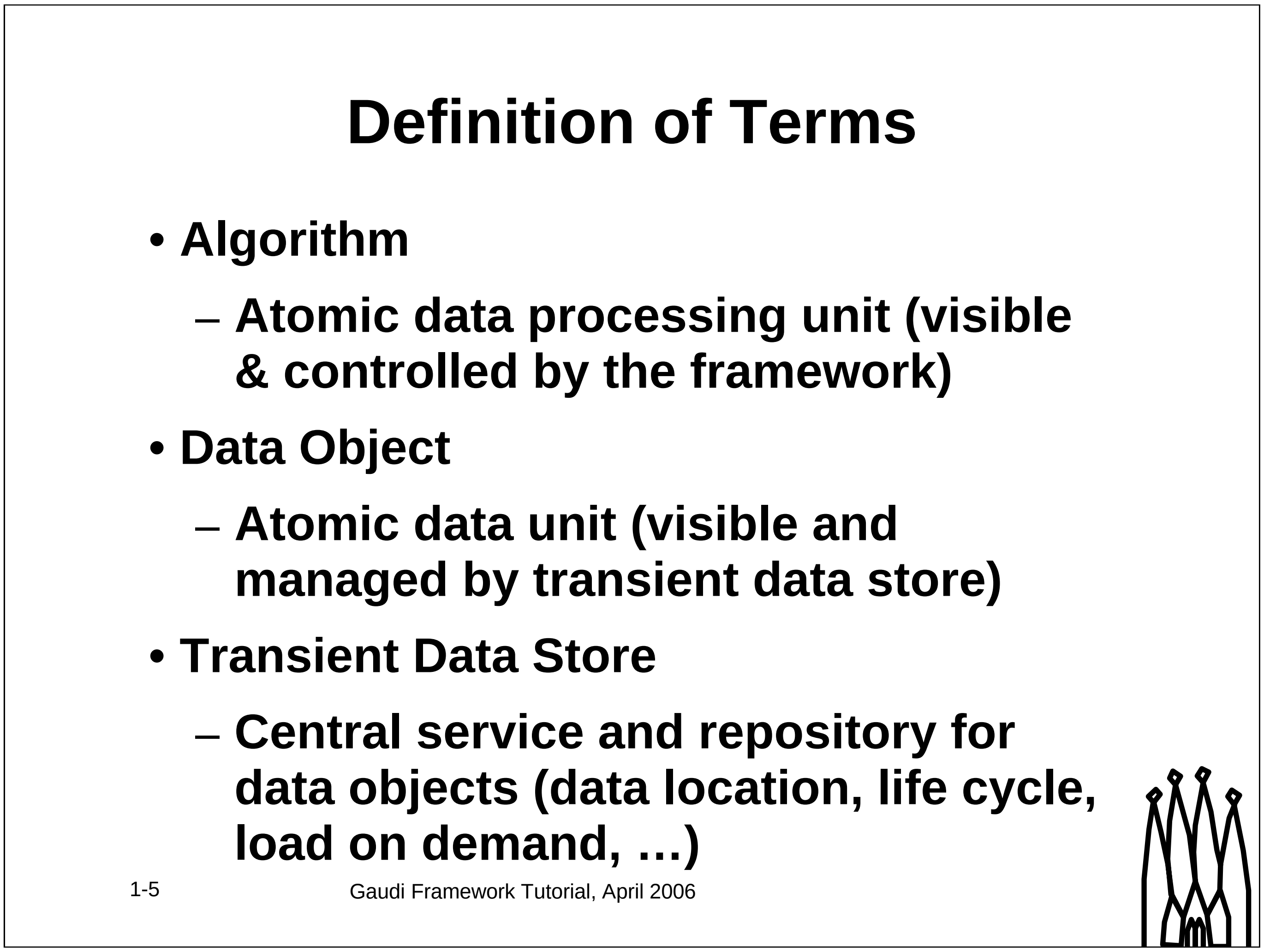


1-4 Gaudi Framework Tutorial, April 2006

Gaudi Object Diagram

The main components of the GAUDI software architecture can be seen in the object diagram in the slide. Object diagrams are very illustrative for explaining how a system is decomposed.

The essence of data processing applications are a set of Algorithms. This is what is shown in the diagram. These Algorithms require the functionality provided by a set of Services. See next slides.



Definition of Terms

- **Algorithm**
  - **Atomic data processing unit (visible & controlled by the framework)**
- **Data Object**
  - **Atomic data unit (visible and managed by transient data store)**
- **Transient Data Store**
  - **Central service and repository for data objects (data location, life cycle, load on demand, …)**

1-5 Gaudi Framework Tutorial, April 2006

Algorithm

The essence of the event data processing applications are the physics algorithms, which are encapsulated into a set of components that we call *Algorithms*. Algorithms implement a standard set of generic interfaces and can be called without knowing what they really do. In fact, a complex algorithm can be implemented by using a set of simpler ones.

Data Object and Transient Store

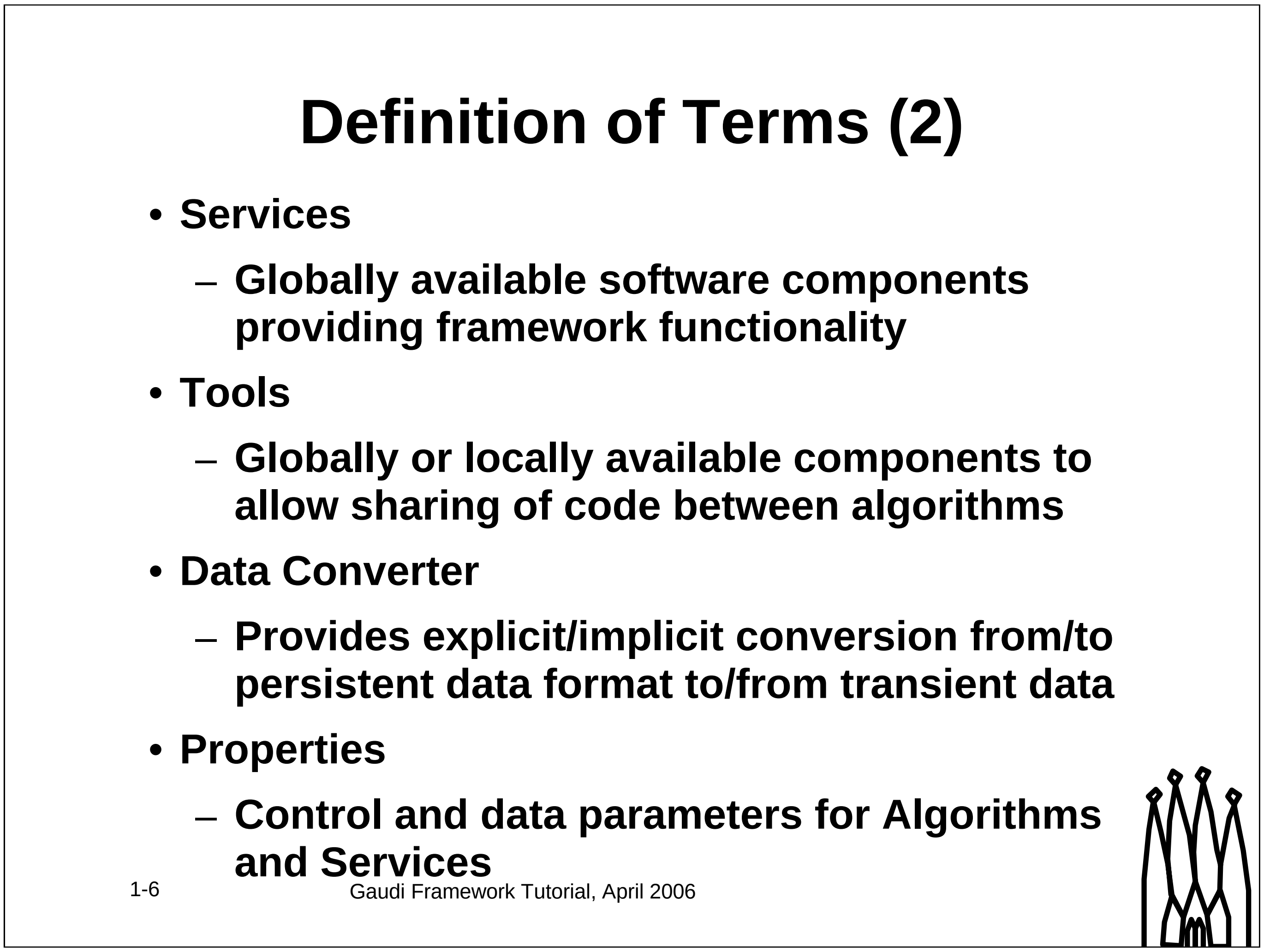
The data objects needed by the algorithms are organized in several transient data stores, depending on the nature of the data itself and its lifetime: event data, detector data, statistical data.. Although the stores behave slightly differently, particularly with respect to the data lifetime (e.g. the event data store is cleared for each event), their implementations have many things in common and are based on a common component.

Mapping to the old FORTRAN terms

•An Algorithm is equivalent to a FORTRAN subroutine that produces new ZEBRA banks from some input banks

•Data Object is equivalent to a ZEBRA bank

•Transient Data Store is equivalent to the ZEBRA common block



Definition of Terms (2)

- **Services**
  - **Globally available software components providing framework functionality**
- **Tools**
  - **Globally or locally available components to allow sharing of code between algorithms**
- **Data Converter**
  - **Provides explicit/implicit conversion from/to persistent data format to/from transient data**
- **Properties**
  - **Control and data parameters for Algorithms and Services**

1-6 Gaudi Framework Tutorial, April 2006

Services

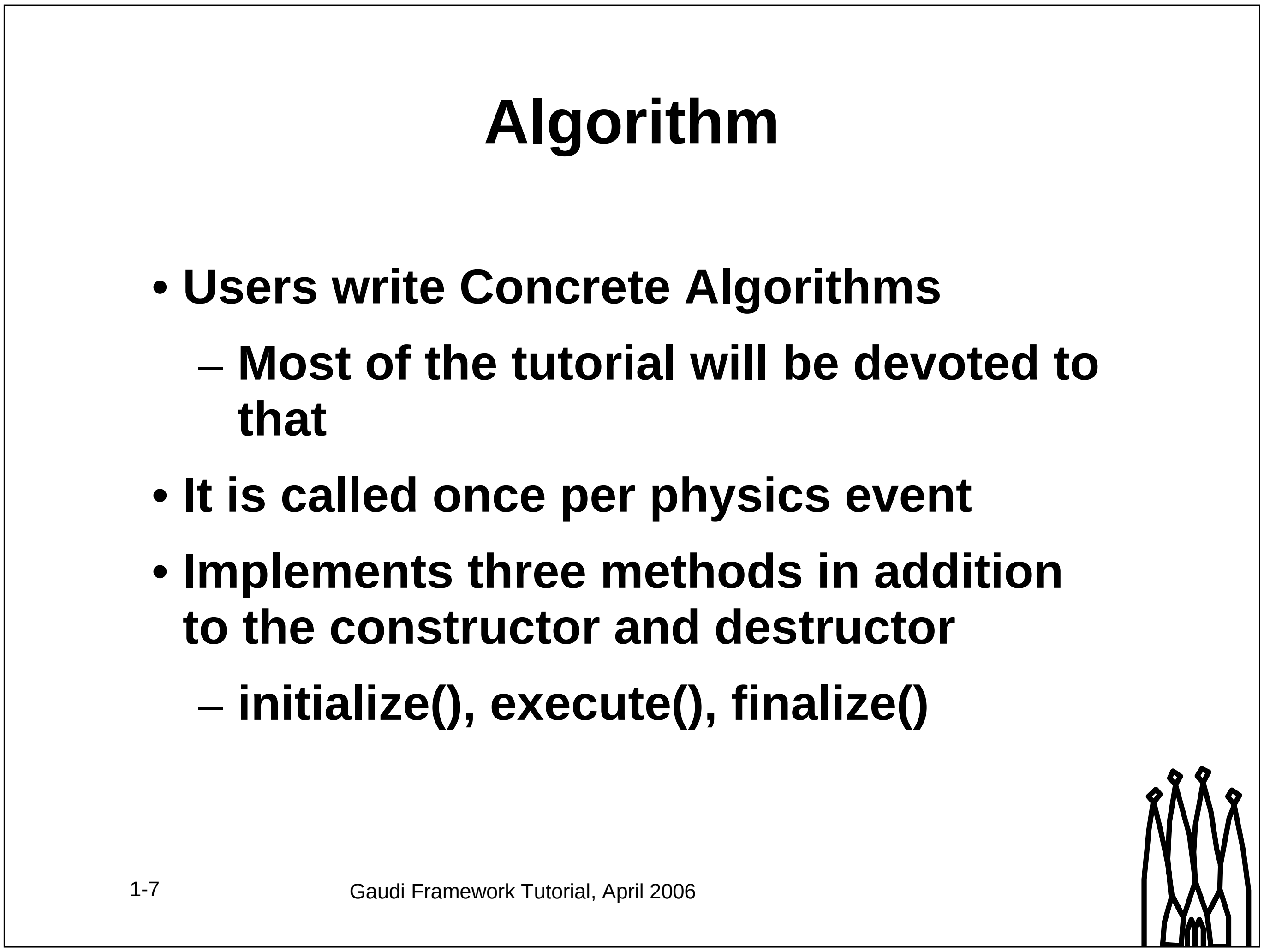
Services are a category of components that should offer all the services directly or indirectly needed by the algorithms. This approach releases the algorithm builder from having to develop the routine software tasks that are typically needed in a physics data processing application. Some examples of services can be seen in the object diagram. They include:

– The services for managing the different transient stores (event data service, detector data service,...) should offer simplified data access to the algorithms.

– Tools can be locally owned by an algorithm or globally available. They are components that allow sharing of code between algorithms (e.g. track extrapolation)

– The different persistency services provide the functionality needed to populate the transient data stores from persistent data and vice versa. These services require the help of specific *converters* which know how to convert a specific data object from its persistent representation into its transient one, or the other way around.

– The job options service, message service and particle properties service.



Algorithm

- Users write Concrete Algorithms
  - Most of the tutorial will be devoted to that
- It is called once per physics event
- Implements three methods in addition to the constructor and destructor
  - initialize(), execute(), finalize()

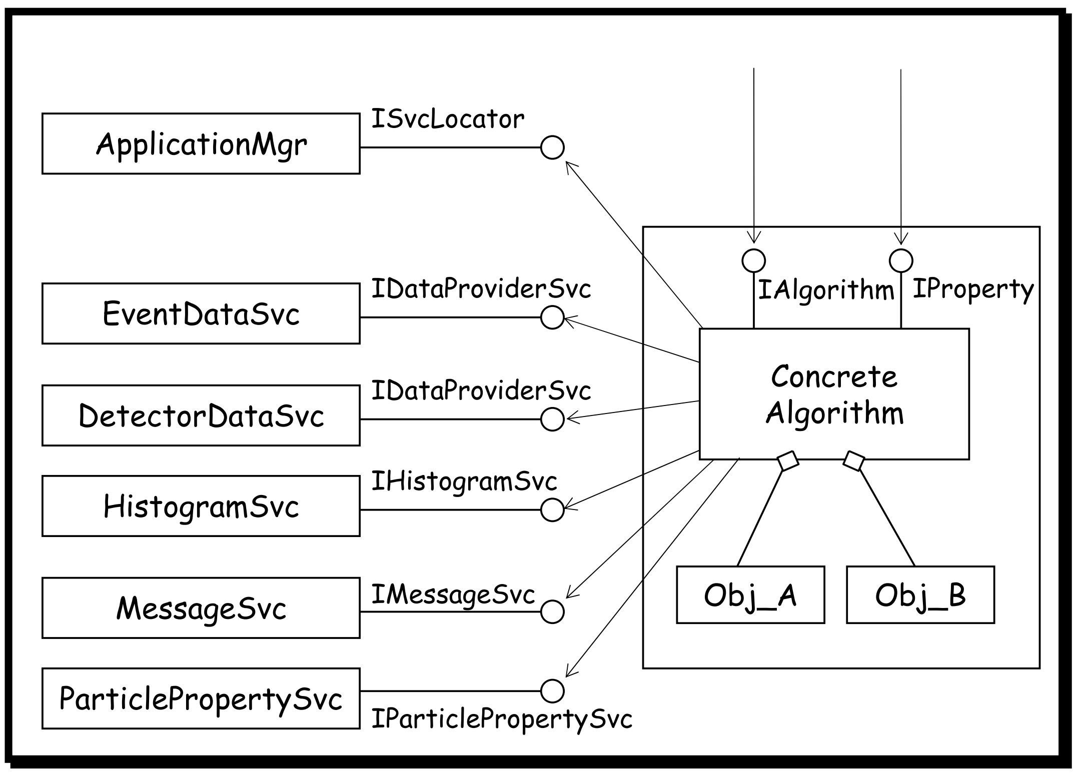
1-7 Gaudi Framework Tutorial, April 2006

Algorithms

Algorithm encapsulates the physics contents of the data processing program. It is the “place holder” foreseen by the framework for the end-user code. This is why most of the tutorial will be devoted to development of algorithms.

Algorithms are scheduled to be executed explicitly once per physics event. If the complexity requires, the algorithm can be made as a composite of a number of sub-algorithms. The complex algorithm schedules explicitly the execution of the sub-algorithms in the proper order to produce the desired results. If an algorithm requires some data that happens to be produced by another algorithm, then the system has to ensure by explicit coding that the algorithms are executed in the correct sequence.

Interfaces

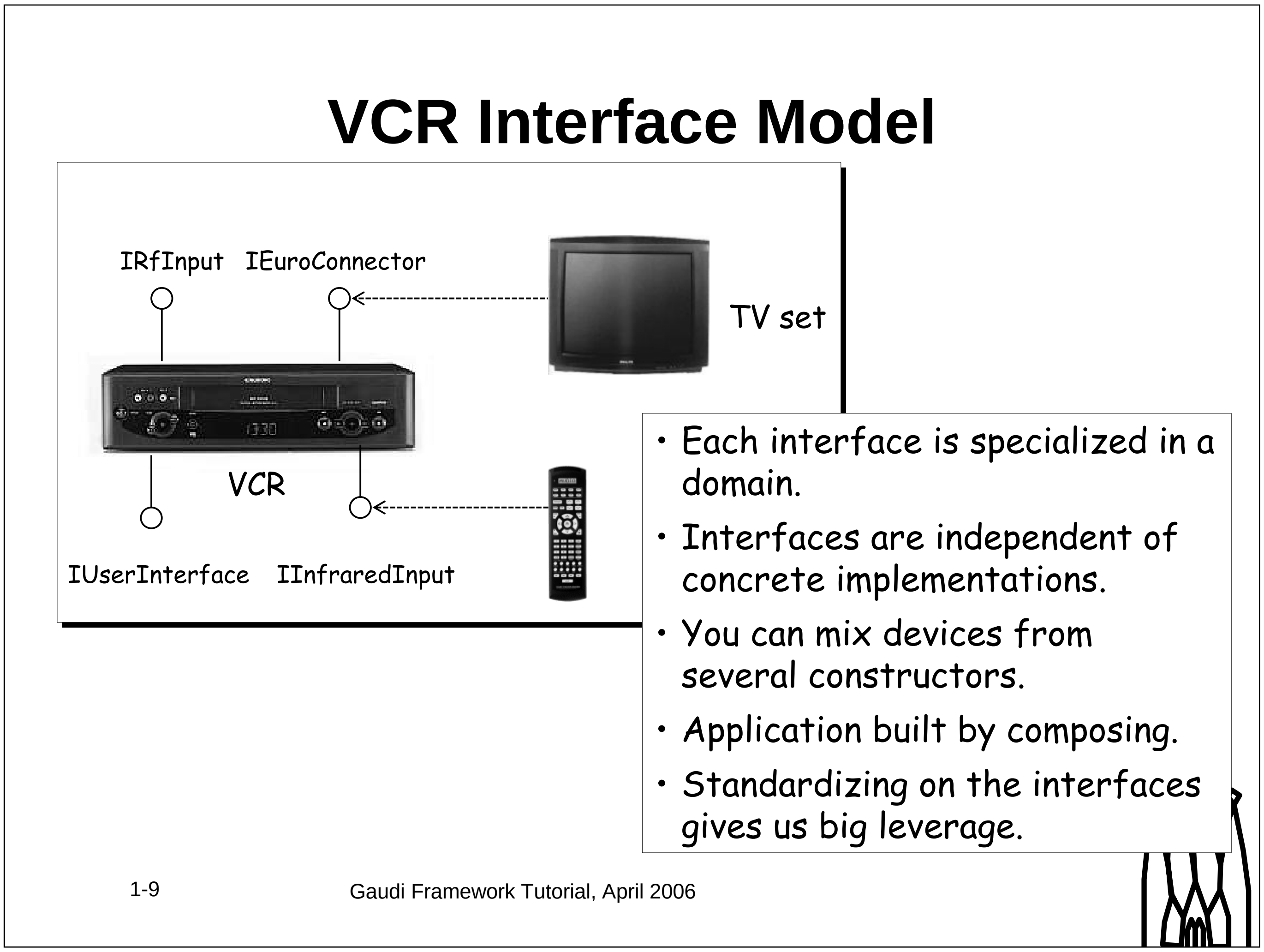


1-8 Gaudi Framework Tutorial, April 2006

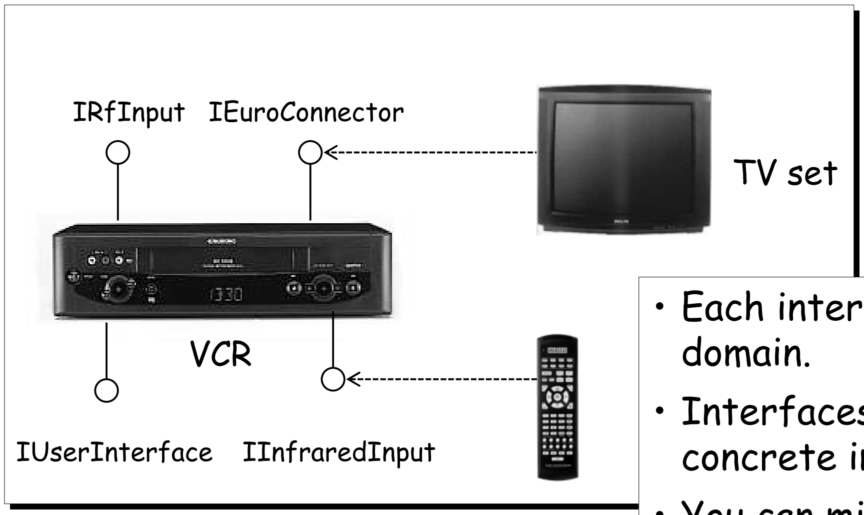
Basic abstract interface concept

This figure illustrates the concept of an Concrete Algorithm implementing a couple of abstract interfaces (IAlgorithm, IProperty) and at the same time “using” a number of other abstract interfaces offering some services (histogramming, messages, particle properties, etc,). The interaction between the algorithm and the services are always through this interfaces. This is the technique that minimizes the coupling between software and allows to be able to replace different implementation without affecting the algorithm clients.

The figure shows the full names of the services (needed e.g. when setting their properties). Methods with shorter name are available in the Algorithm base class to access these services (e.g. eventSvc(), detSvc(), histoSvc(), msgSvc())



VCR Interface Model

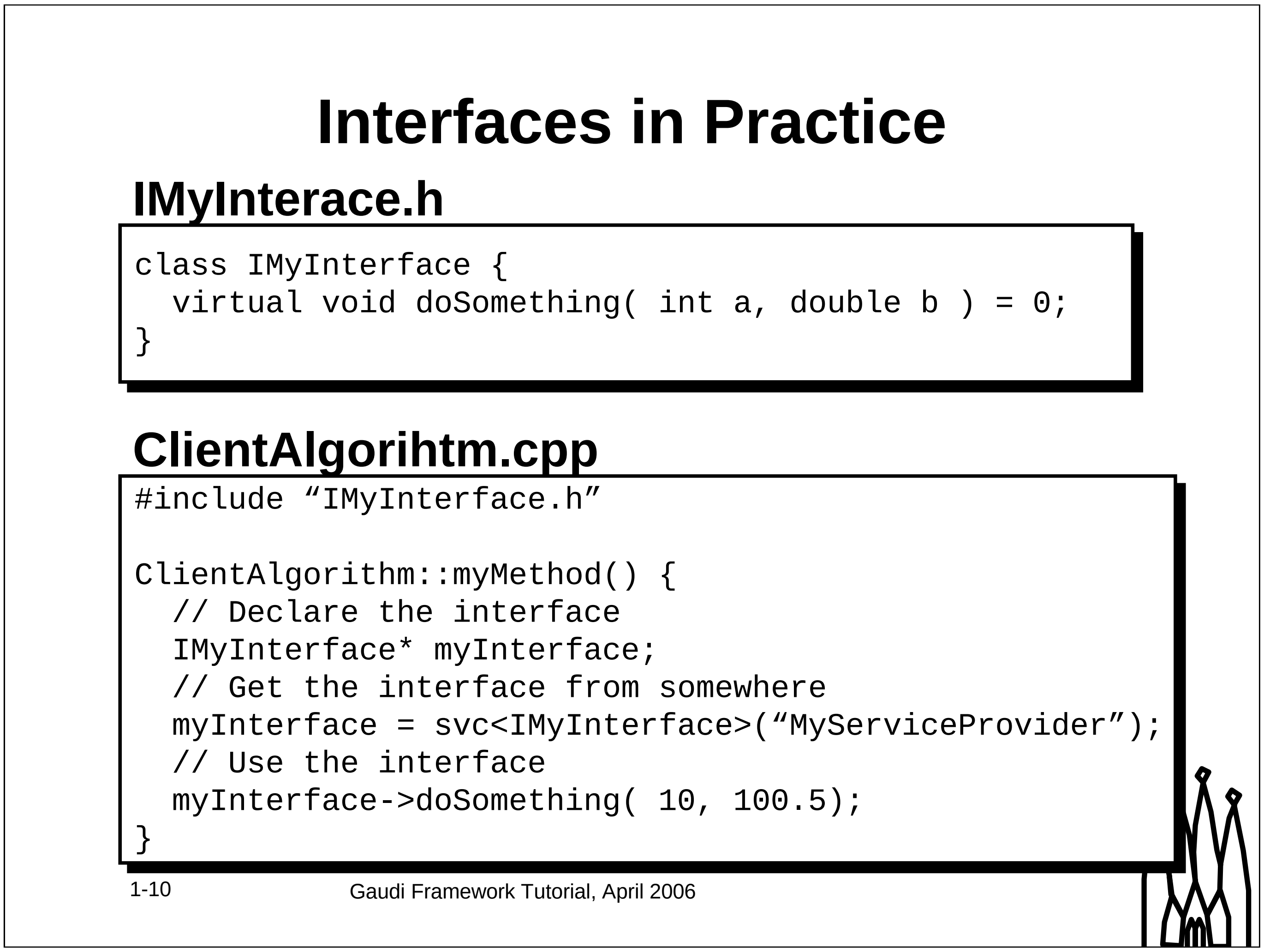


- Each interface is specialized in a domain.
- Interfaces are independent of concrete implementations.
- You can mix devices from several constructors.
- Application built by composing.
- Standardizing on the interfaces gives us big leverage.

1-9 Gaudi Framework Tutorial, April 2006

VCR Interface Model

This analogy illustrates the building of a system from a number of components connected by their interfaces. It clearly shows the advantages of standard interfaces and the possibility to evolve the system by replacing some of the components without having to re-do the rest of the system.



Interfaces in Practice

IMyInterace.h

```
class IMyInterface {
  virtual void doSomething( int a, double b ) = 0;
}
```

ClientAlgorihtm.cpp

```
#include “IMyInterface.h”

ClientAlgorithm::myMethod() {
  // Declare the interface
  IMyInterface* myInterface;
  // Get the interface from somewhere
  myInterface = svc<IMyInterface>(“MyServiceProvider”);
  // Use the interface
  myInterface->doSomething( 10, 100.5);
}
```

1-10 Gaudi Framework Tutorial, April 2006

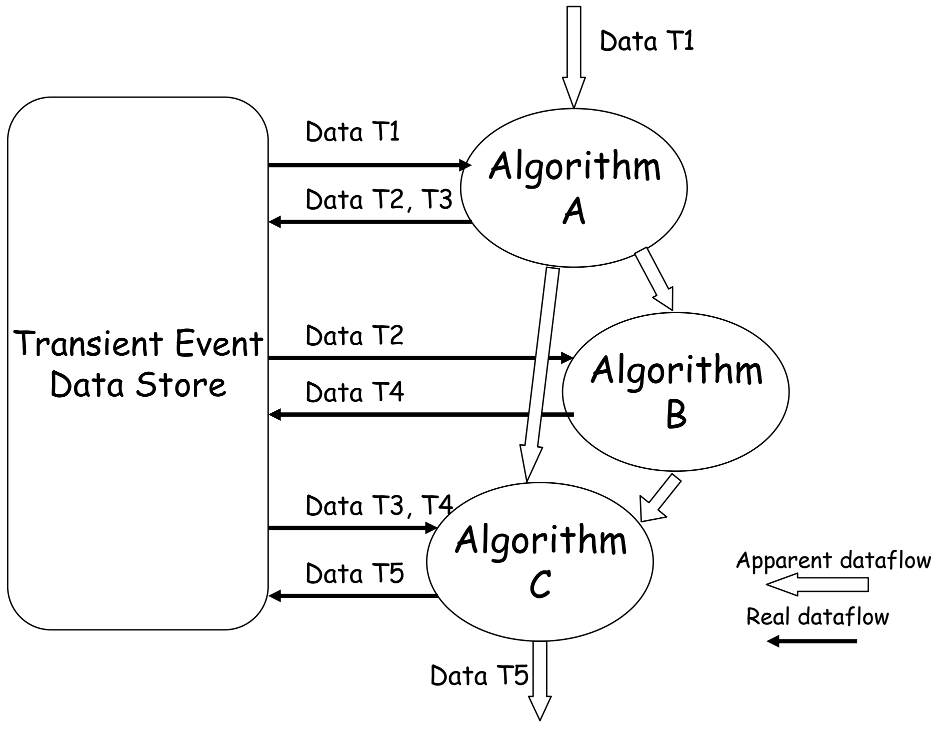
Interfaces in Practice

The declaration of the interface (IMyInterface.h) contains a number of abstract methods (= 0). The client of the interface does not know who is implementing the interface, it has no necessity to know it. It needs to include only the interface definition (this is the contract between the client and the implementation)

Notice that an interface (class with abstract methods) can not be instantiated. You can not call “new IMyInterface”, therefore you need to have a mechanism to instantiate objects implementing the interface indirectly. This is one of the functionalities offered by the framework.

The templated svc function in the example locates the service whose name is “MyServiceProvider”, and returns a pointer of type IMyInterface*. The algorithm can then use this service. Note that the algorithm only has a pointer to the service, which is guaranteed to be valid. It does not *own* the service so it does not have to worry about deleting it.

Algorithm & Transient Store



1-11 Gaudi Framework Tutorial, April 2006

Algorithm & Transient Store

In order to minimize the coupling between algorithms, the transient data store acts a “black-board” between algorithms. An Algorithm does not need to know from where the data has been produced, it only knows what data it requires and what data will be producing.

As in shown in the viewgraph, we can create “data-flows” using a number of Algorithms known only, their inputs and outputs, by scheduling them in the adequate order.

Not e that for this to work, a contract must be made between algorithms, that a given algorithm cannot modify data that is already on the Transient Store. In general, when an algorithm puts data on the transient store, it relinquishes ownership of that data. The data is then owned by the TES and should be considered as read only – the TES will eventually take care of deleting it.

Gaudi Services

- JobOptions Service
- Message Service
- Particle Properties Service
- Event Data Service
- Histogram Service
- N-tuple Service
- Detector Data Service
- Magnetic Field Service
- Random Number Generator
- Chrono Service
- (Persistency Services)
- (User Interface & Visualization Services)
- (Geant4 Services)

Gaudi Services

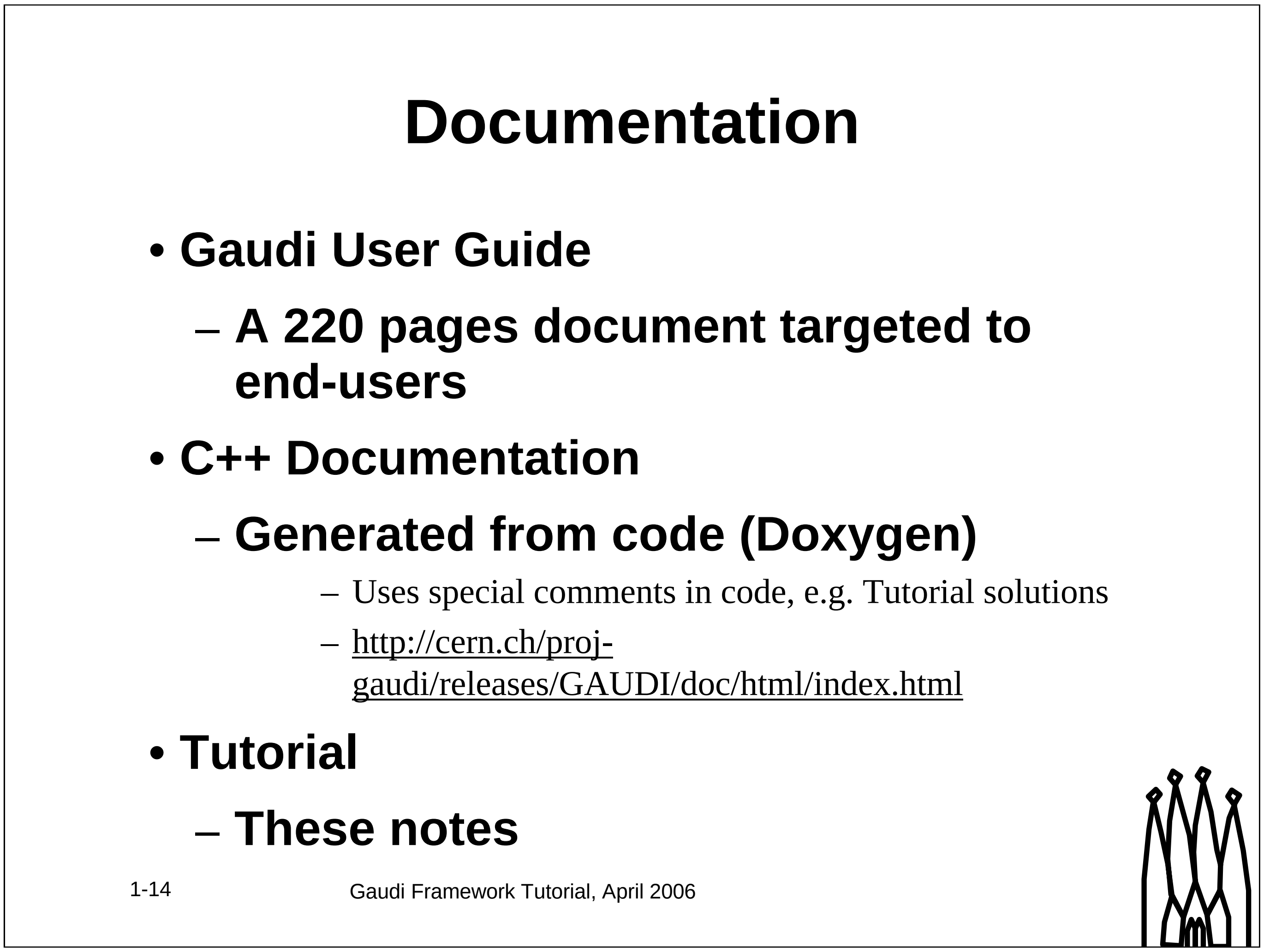
Open ended list of available services in Gaudi. The services in the top of the list will be the ones that will be used and exercised during the tutorial. Many of the existing services are not really used directly by the Algorithms implementing a physics algorithm but they are used by the Framework itself in implementing some of the functionalities (persistency, visualization, interactivity, etc.)

Gaudi Product Sheet

- **Current release**
  - **v18r3 (March 06)**
- **Supported Platforms**
  - **Scientific Linux (CERN) 3 & gcc 3.2.3**
  - **Windows 2000,XP & VisualC++ 7.1**
- **Web address.**
  - **http://cern.ch/proj-gaudi/welcome.html**

Gaudi Product Sheet

Note that a number of experiment neutral packages are managed from an independent project area (repository, web, etc,). These packages are shared between experiment (LHCb, ATLAS, HARP, etc.). The experiment neutral Gaudi is located in http://cern.ch/proj-gaudi



Documentation

- **Gaudi User Guide**
  - **A 220 pages document targeted to end-users**
- **C++ Documentation**
  - **Generated from code (Doxygen)**
    - Uses special comments in code, e.g. Tutorial solutions
    - http://cern.ch/proj-gaudi/releases/GAUDI/doc/html/index.html
- **Tutorial**
  - **These notes**

1-14 Gaudi Framework Tutorial, April 2006

Documentation

All this documentation is accessible from the main Gaudi web page. Not that the user guide has not been updated for two years, it is planned to update it “soon”

The doxygen documentation above is limited to the Gaudi (experiment neutral) classes. LHCb specific classes (e.g. the event model) are documented within the LHCb application projects, e.g.

http://cern.ch/LHCb-release-area/LHCB/doc/html/index.html

In addition to the mentioned documents, many books cover more standard aspects of the software development (C++ language, O-O Design, etc,). A selection of such books are available in the small “LHCb Computing Library” or can be purchased from in the CERN IT Bookshop.