Use-Case-Driven Design of Atlas Software

Atlas Software Week

1 September, 1999 @CERN

Katsuya Amako

(KEK)

A mail to subsystem software coordinators

On 5th July, Norman McCubbin sent out a mail to the subsystem software coordinators saying:

> “At the first meeting of the Architecture Task Force last week we agreed that input on various topics from the sub-system s/ware communities was very important.
>
> The first topic on which your input would be appreciated is that of 'Use-cases', or, more informally, 'What do you want to be able to do' with the Architecture.
>
> May I ask you please to have let us (N.McCubbin@RL.AC.UK and Katsuya.Amako@kek.jp) have your and your group's first thoughts on this topic by the end of Tuesday 13th July.”
>
> ……..
> ……..

Responses from the subsystem software coordinators

Responding to Norman's request, we received answers from the following people.

- **Calorimeter -** J. Beck Hansen, J. Collot, M. Leltchouk, P. Loch, J. Pinfold,

S. Rajagopalan, J. Schwindling, S. Simion,

- **Inner Detector** - Dario Barberis (also for test-beam software)
- **Muon Detector** - Gilbert Poulard
- **Jet reconstruction** - M.Bosman. P.Loch

We thank all of you. These inputs are truly the starting point to proceed the **use-case-driven** design of Atlas software.

Use case and use case model

- **What is use case**?
  - A use case describes a way the user uses a software system.

Or more rigorously,

  - Use cases capture functionality of the software system as seen by users.

    i.e. it is a kind of user's requirements document.

  Here
  - the software system means Atlas software system,
  - the users mean all Atlas members.

- **What is use case model**?
  - It consists of use case diagrams and statements which helps the users and developers agree on how to use a software system.
  - The model is described in plain English with a simple diagram notation. A user can easily understand the model without any special knowledge.
  - A user doesn't need to know more than this use case model. The realization (=implementation) of use cases have to be done by developers.

What is use-case-driven?

Using use cases as a **primary artifact** to design and implement the software system which satisfies the user's requirements.

➔ This is called "use-case-driven".

The points are

- A software developer need to design and implement software which is satisfied by users.
- Use cases specify what functionality users want to see in a software system.

Although this idea is motivated by a very pragmatic view (= attract users and earn money or the software company will go bankrupt), it can be a good principle also in HEP.

➔ I believe this pragmatic approach is important in designing the Atlas software system.

Who are users and who are developers?

In a HEP experiment physicists are sometimes software users and sometimes are software developers.

- This point was already emphasized by Fabiola Gianotti in her talk yesterday.

So we need to be very careful when we say “user” or “developer”.

- We need to define clearly the meaning of “user” before creating our use case model.
  - In Unified Modeling Language (UML), a user is called an ‘actor’.In UML actors have to be defined before creating a use case model.
- Considering actors in the Atlas software system, we use the responses provided by subsystem software people.

Domain Area of Atlas Software System

- **Domain area**

  When you read the responses from detector subsystem software people, you realize that they are talking about “use-cases” of almost all activities in Atlas data processing and analysis in online and offline.

  Therefore the **domain area** of software we need to design encompasses all activities of Atlas data processing and analysis in offline. It also includes event filtering and physics monitoring in online.

  Definition: **Domain area**

  > An area of knowledge or activity characterized by a set of concepts and terminology understood by **practitioners** in that area.

  practitioners = detector subsystem people and all Atlas people who will involve in data processing and analysis

  practitioners [1] “software people” (=software developers)

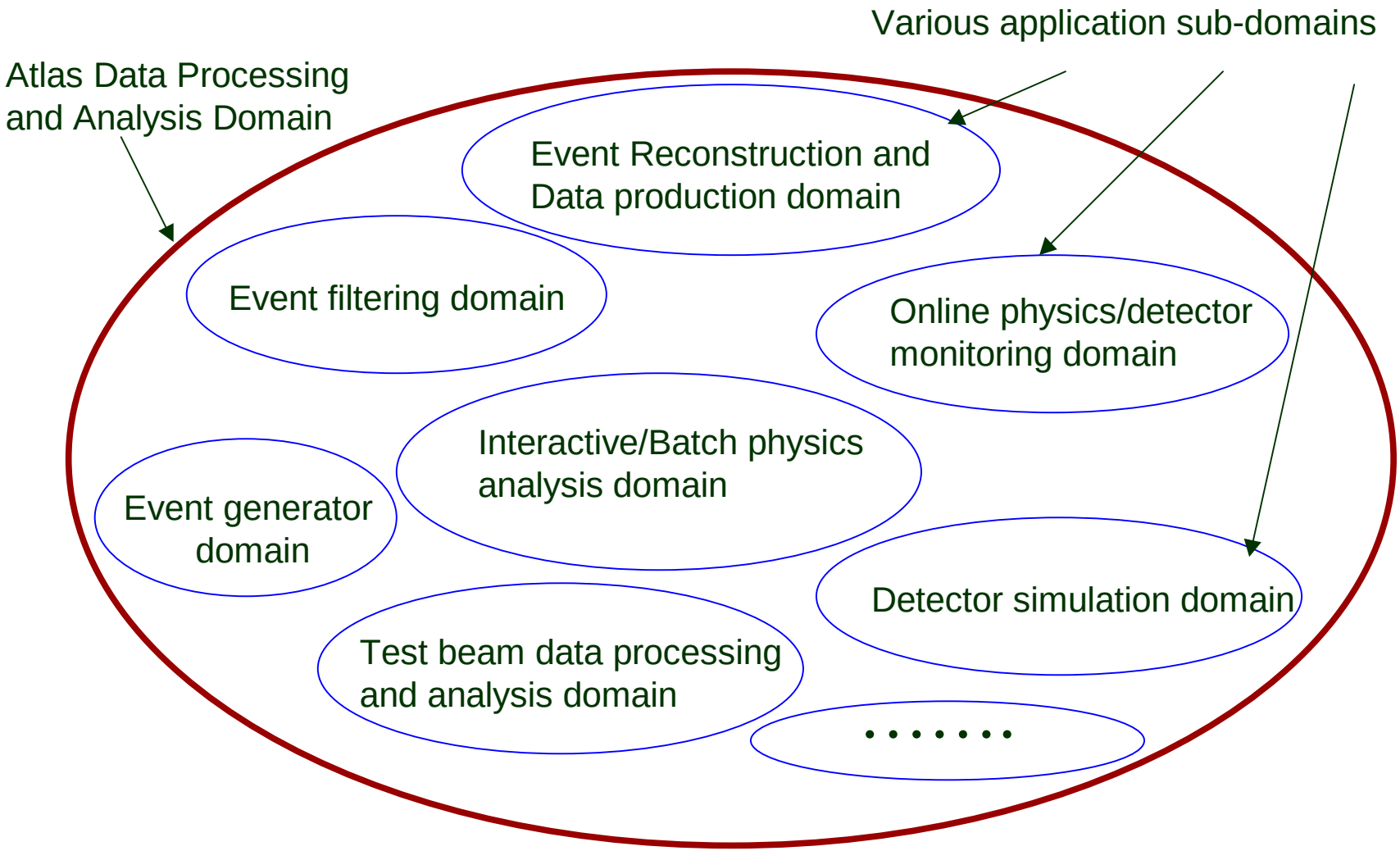
Domain Decomposition of Atlas Software System - 1

- **Domain Decomposition**

  From our experience in HEP data processing and analysis, we know that we can decompose our problem domain into sub-domains: (*we need more elaboration for appropriate decomposition*)

  - Online physics/detector monitoring domain
  - Event Reconstruction and data production domain
    - Data calibration
    - Reconstruction parameter tuning
    - Development/Refinement of reconstruction algorithms
    - ……..
  - Event filtering domain
    - Online event filtering
    - Offline event filtering
  - Interactive/Batch physics analysis domain
  - Event generation domain
  - Detector simulation domain
  - Test beam data processing and analysis domain
  - …….

Domain Decomposition of Atlas Software System - 2



Domain Decomposition of Atlas Software System - 3

Of course activities in these sub-domains are **NOT AT ALL** independent. For example, across various application domains, people want to use

- common tools (event display, histogramming, curve fitting, etc),
- common data presentation,
- common geometrical data source,
- common data storage/retrieve procedure,
- ……

Therefore it is clear that we need a common building blocks to design the software system in each sub-domain.

[Note]

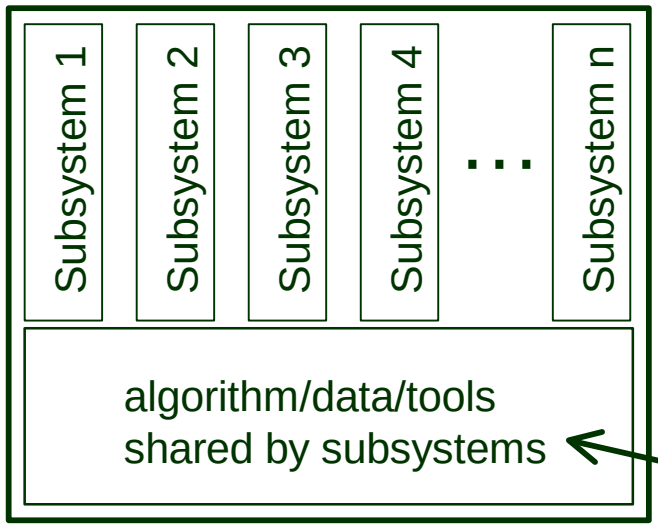
In the above list, some need to be unique across applications, while other are **not** necessary be unique. Example for tools which are not necessary be unique are histogramming tool, graphics tool, etc. People probably want to use their own favorite ones.

- ➔ The common building blocks should be designed independently to any particular tool.
- ➔ We, many physicists, hate a big monopoly like MicroSoft.

  (I'm writing this slide using MS-PowerPoint, though I hate MS-Office!)

Layered Structure in the Atlas Software System

So natural structure of the Atlas software system is like this:



Let's call this global structure as
Atlas Software System

For example,

Subsystem 1 = Event generator system
Subsystem 2 = Detector simulator system
Subsystem 3 = Data production system
……..

Let's us call this as
Atlas Common Framework
(I don't want to get involved in the discussion of what is a framework. Please take as it is defined in here.)

Strategy to design Atlas software system - 1

First of all we need to design the Atlas common framework by the use-case-driven approach.

So let's go back to the original question, i.e.

Who are users (= actors) of this common framework?

Here actors are not necessary be human being, but the software in subsystems.

For example,

- the detector simulator wants to get geometrical information from the common framework,
- the tracking finding package in the data production software wants to get geometrical information from the common framework, etc.

The first goal of the ATF (in my opinion, not yet discussed) is to establish the **use case model** of the Atlas common framework.

Strategy to design Atlas software system - 2

To create the use case model, we need serious interactions with

- people who are designing /implementing subsystem software,
- people from detector subsystems,
- people form physics subgroups.

[Note]

Rigorously the responses subsystem software coordinators provided us by our request are not use cases but are candidates for use cases.

- ➔ A collection of use case candidates is called the **feature list**. All responses from sub-detector people are currently compiled in this feature list.
- ➔ Using statements in the feature list, we will elaborate them into use cases. The serious interactions with people mentioned above includes this elaboration process.

In a modern software engineering, to create a use case model means to establish a requirement document.

In any design, establishing a meaningful requirement document is a huge and difficult task - this is true for both hardware and software.

- ➔ Though it is an essential step in designing the ATLAS software.

Strategy to design Atlas software system - 3

Actually establishing a use case model is just a first step in the process to develop the Atlas software system.

In the 1st ATF meeting (2nd July), I proposed the following global software development process:

1. To write a requirement document and to create 'use-case' of the software system which will be used for the Atlas data processing.

   ← Today, I just concentrated this step in my talk.

2. To do object-oriented analysis.

3. To create high-level class/object models and interaction/collaboration diagrams.

4. To define packages and their relations/dependencies and to create a package diagram.

5. To define signatures of major methods of high-level classes so that further details of design can be done in a wider community.

6. To do object-oriented design in the wider community.

7. To finalize class/object models, interaction/collaboration diagrams. Also define signatures of major methods in classes found in object-oriented design.

Strategy to design Atlas software system - 4

Actually the software development process I mentioned is not a single straight line path but it is an

- **iterative and incremental process**

in sense of

Booch methodology

or more recent one,

The Unified Software Development Process (USDP) by I. Jacobson, G. Booch and J. Rumbaugh.

To establish a software development process is essential in creating a huge and complex software system as in Atlas.

- ➔ Geant4 is a relatively small software system comparing to Atlas, though it was essential to have its own software development process for a successful international collaboration.
- ➔ Geant4 used the Booch methodology for establishing its software development process.

Strategy to design Atlas software system - 5

[Note]

- Although the software development process (I.e. software methodology) has fundamental importance in Atlas software development, most Atlas members don't need to know what it is.

  For example, suppose you implement your own physics analysis program, or track fitting program,

  - you do **not** need to know the software process,
  - the existence of the software process doesn't affect at all to your implementation,
  - only thing you need to know is a very minimum knowledge of C++ programming.

    [Note to the Note]

    In Geant4, all physics processes (the heart of Geant4) were done by physicists who didn't know C++ at all at the beginning. There was no difficulty for them to start coding of their physics processes once they learned a minimum C++ knowledge - no extensive C++ course nor reading the thick Strroustrup's book.

Goal and product (in the term of the ATF)

- Goal

  At the end of ATF we want to create the analysis model of the Atlas common framework (no iteration, just the 1st trial)

- Products (we haven't discussed yet in the ATF, but …..)

  Design documents of the **use-case model and analysis model** of the Atlas common framework. Unified modeling language will be used in the design document.

  [Note]

  Although UML will be used, what most people in Atlas need to understand is the use-case model, which are described in plain English and simple diagrams.

People (in the term of the ATF)

- Katsuya Amako
- Laurent Chevalier
- Andrea Dell'Acqua
- Fabiola Gianotti
- David Roussea
- Craig Toll
- Valerio Versesi

Real code?!

A natural question you may want to ask is:

> When can I see and play with the C++ codes of the Atlas common framework?

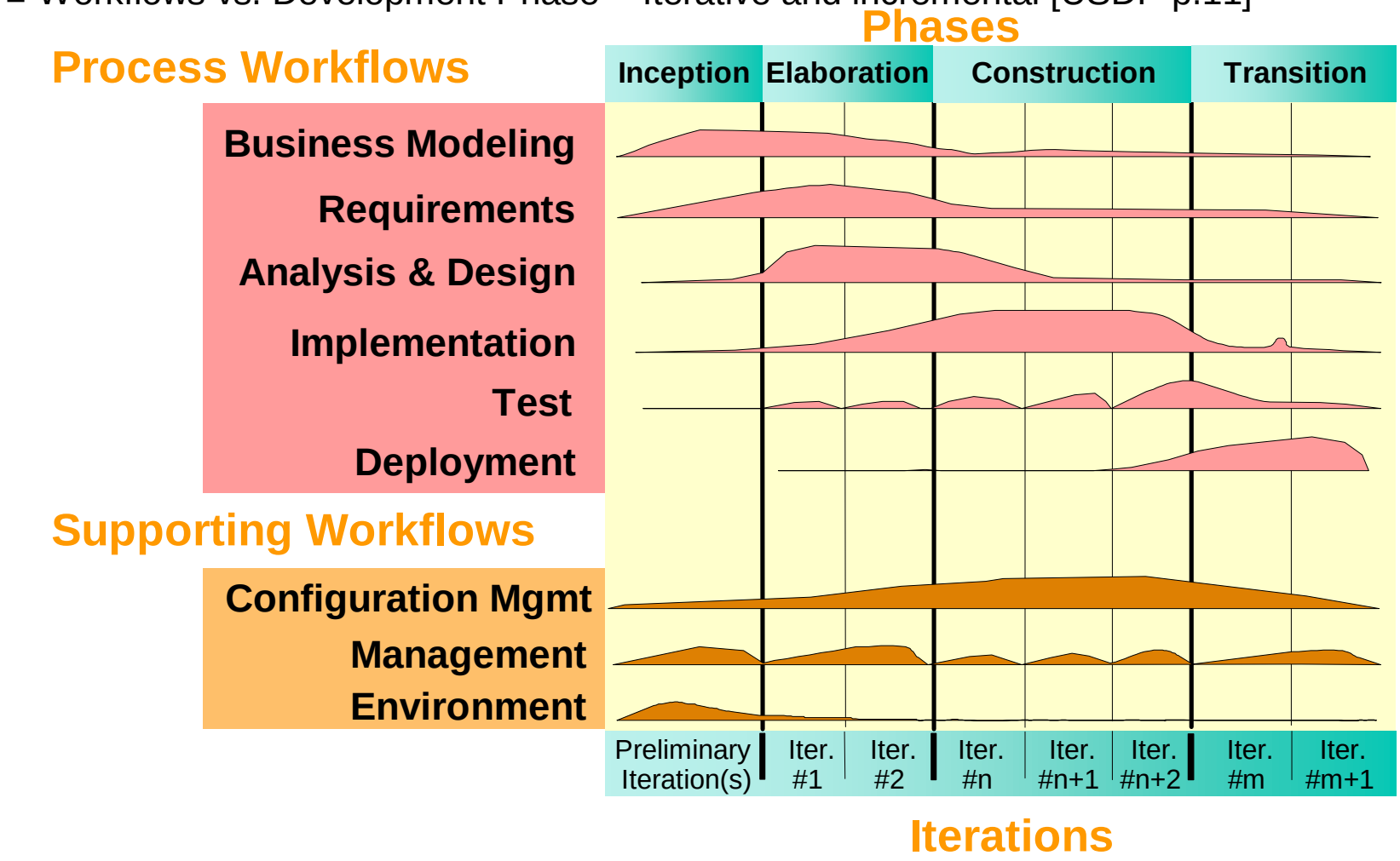
Well, the implementation is the out of scope of the ATF, though please remember that

> the heart of **iterative and incremental process** is to provide a test implementation of code at the really early stage of the development process, i.e. the end of the 1st iteration of the cycle (i.e. requirement-analysis-design).

➔ See the next figure.

Software Development Process: USDP

- Workflows vs. Development Phase - Iterative and incremental [USDP p.11]



Summary

■ Using the responses from sub-detector software coordinators, ATF started to create a use case model of the Atlas common framework.

■ This work requires further collaboration with sub-detector software coordinators. Also it requires interactions with physics groups.

■ The use-case-driven approach is important in designing the Atlas software system.

■ To create a use case model is just a first step of iterative and incremental development of the Atlas software.

■ It is also important to establish our own software development process adapting ‘Unified Software Development Process’.

[Note]

The above summary reflects my own view of the ATF work - we will more in the next ATF meeting.