Status of USDP Activities

Atlas Software Week

10 May, 200 @LBL

Katsuya Amako
(KEK)

Major goal of USDP work

- To create major ***use-cases*** of ***Atlas common framework***, which is defined in the 'Report from the Atlas Architecture TaskForce'.

- These use-cases will be used when Atlas compares the functionality provided by the Gaudi with Atlas' own framework requirements.

[Note]

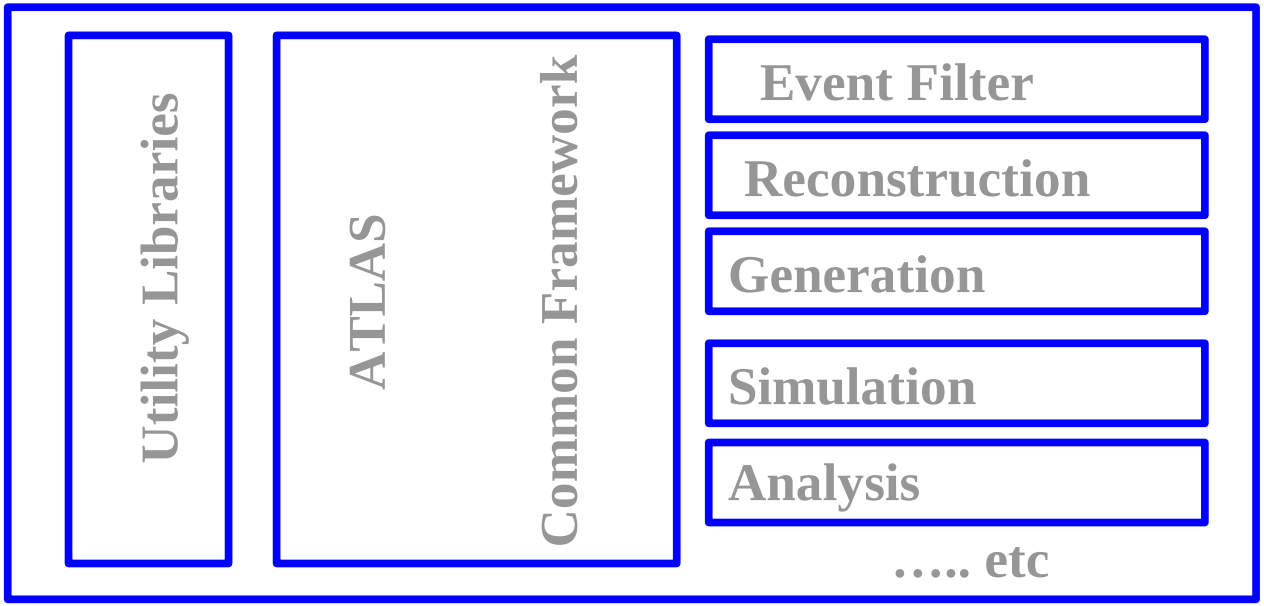
The USDP work in Architecture Team is limited only to prepare use-cases for the Atlas common framework. No OOA/OOD will be pursued.

(Decision by A-team on 10 March, 2000)

Atlas Common Framework

What is “framework”?

- A collection of classes that provide a set of services for particular application domain; a framework thus provide a number of individual functionalities and mechanisms that the user can use or adapt to build an application software.Frameworks may actually be domain-neutral, meaning that they apply to a wide variety of applications. [Booch p.327]



What is ‘Use case’?

- A software system is brought into existence to serve its users. Therefore, to build a successful system we must know what its prospective users want and need.
- The term ‘user’ refers not only to human users abut to other systems. The term ‘user’ represents someone or something that interacts with the system.
  - In USDP, they use the term ‘actor’ instead of ‘user’.
- What is ‘use-case’?
  - An interaction of of actors to a software system.
  - A use case is a piece of functionality in the system that gives a user a result of value.
  - User cases capture functional requirements.

(From the USDP book, p.5)

Workflow

- Requirement capturing steps by use-cases in USDP

  1) Finding actors

  2) Finding use cases.

  3) Briefly describing each use case.

  4) Describing more detail of each use case - writing use case specification

(From the USDP book, p.145)

Workers

- Core workers
  - Katsuya Amako
  - Chris Day
  - Massimo Marino
  - Gilbert Poulard
  - David Rousseau
- Inputs from subsystem are essential
  - The core workers play interface to subsystem software people.

Finding primary actors from framwork

- Detector description provider
- Magnetic field provider
- Online calibration and alignment
- Event filter
- Offline calibration and alignment
- Offline event reconstruction
- Offline event streaming
- Physics analysis
- MC Event generator (4-vector)
- Detector simulator
- Visualization
- …..

Finding use cases for framwork - 1: Bottom up

- To start from specific services, for example
  - Use subsystem event data
    - store/retrieve/use silicon data
    - store/retrieve/use Liq.Ar data
    - …….
  - Use subsystem detector description
    - store/retrieve/use silicon detector description
    - store/retrieve/use Liq.Ar detector description
    - .....
- And generalize them for common framework
- Strategy
  - To adapt Johann Collot's extensive works on liq.Ar.
    ➔ To generalize it for common framework.

Finding use cases for framework - 2: Top down

- **To start from application view, for example**
  - Use muon data analysis program
  - Use liq.Ar data analysis program
  - Use testbeam data analysis program
  - Use full simulation program
  - ….
- **Strategy**
  - Interaction with subsystem people
    - Tile
    - Liq.Ar
    - Tracker
    - Muon
  - Re-study on existing documents (feature list and other docs)

Use case specification: Template by Chris Day - 1

1. Use Case ID - *Assigned by System*

2. Use Case Name - *[Replace all text in brackets with your text]*

2.1 Brief Description

[The description should briefly convey the role and purpose of the use case. A single paragraph should suffice for this description.]

3. Initiating Actor

[An Actor who expects something of value to be supplied by the use case initiates it. Identify the recipient of that value here.]

4. Value Delivered

[Describe the value delivered to the Initiating Actor here.]

5. Participating Actors

[Other Actors may act as suppliers for this use case. Identify them here.]

6. Pre-Conditions

[Pre-condition (of a use case) is the state of the system that must be present prior to a use case being performed.]

6.1 < Pre-condition One >

Use case specification: Template by Chris Day - 2

7. Flow of Events

7.1 Basic Flow

[This use case starts when the actor does something. An actor always initiates use Cases. The use case should describe what the actor does and what the system does in response. It should be phrased in the form of a dialog between the actor and the system.

The use case should describe what happens inside the system, but not how or why. If information is exchanged, be specific about what is passed back and forth. For example, it is not very illuminating to say that the Actor enters customer information; it is better to say the Actor enters the customer's name and address. A Glossary of Terms is often useful to keep the complexity of the use case manageable; you may want to define things like customer information there, to keep the use case from drowning in details.

Simple alternatives may be presented within the text of the use case. If it only takes a few sentences to describe what happens when there is an alternative, do it directly within the flow of events section. If the alternative flows are more complex, use a separate section to describe it. For example An Alternative Flow describes how to describe more complex alternatives.

A picture is sometimes worth a thousand words (though there is no substitute for clean, clear prose). If it improves clarity, feel free to paste graphical depictions of user interfaces, process flows, or other figures into the use case to improve its clarity. If a flow chart is useful to present a complex decision process, by all means use it! Similarly for state-dependent behavior, a state-transition diagram often clarifies the behavior of a system better than pages upon pages of text. Use the right presentation medium for your problem, but be wary of using terminology, notation or figures that your audience may not understand. Remember that your purpose is to clarify, not obscure.]

Use case specification: Template by Chris Day - 3

7.2 Alternative Flows

7.2.1 < First Alternative Flow >

[More complex alternatives should be described in a separate section, which is referred to in the basic flow of events section. Think of the alternative flow sections like alternative behavior - each alternative flow represents alternative behavior (many times, because of exceptions that occur in the main flow). They may be as long as necessary to describe the events associated with the alternative behavior. When an alternative flow ends, the events of the main flow of events are resumed unless otherwise stated.]

7.2.1.1 < An alternative sub-flow >

[Alternative flows may in turn be broken down into sub-sections if it improves clarity]

7.2.2 < Second Alternative Flow >

[There may be, and most likely will be, a number of alternative flows in a use case. Keep each alternative separate to improve clarity. Using alternative flows improves the readability of the use case, as well as preventing use cases from being decomposed into hierarchies of use cases. Keep in mind that use cases are just textual descriptions, and their main purpose is to document the behavior of a system in a clear, concise and understandable way.]

Use case specification: Template by Chris Day - 4

8. Invariant Conditions

[Invariant conditions (of a use case) are a list of possible states the system must preserve while enacting a use case.]

8.1 < Invariant Condition One >

9. Post-Conditions

[Post-conditions (of a use case) are a list of possible states the system can be in immediately after a use case has finished.]

9.1 < Post-condition One >

10. Special Requirements

[A Special Requirement is typically a non-functional requirement that is specific to a use case but is not easily or naturally specified in the text of the use case's event flow. Examples of special requirements include legal and regulatory requirements, application standards, and quality attributes of the system to be built, including usability, reliability, performance or supportability requirements. Additionally, other requirements such as operating systems and environments, compatibility requirements, and design constraints should be captured in this section.]

10.1 < First special requirement >

11. Remarks

[Miscellaneous information not belonging to functional or non-functional requirements but related to the use case.]

Use case specification - Web tool by Chris Day

- Interactive user interface via web to write a use-case will be available.
  - Change history of a use-case will be automatically kept
- Use case web page will be provided.

Example of Use case specification

- Generalization of Johann Collot's use-cases
  - LAr-UC-1
  - Storage of Test Beam Event Collection

Use case specification: Generalization of LAr-UC-1 (a)

1. Use Case ID - ***Assigned by System***

2. Use Case Name - **Store an event collection**

2.1 Brief Description

The actor creates a collection of events (event collection) and stores to the *Atlas Common Event Data Store*.The purpose of this action is to secure the event collection created by the actor, and to assure a long term access.

3. Initiating Actor

Online calibration and alignment, On-line event filter, Off-line Calibration and Alignment, Off-line event reconstruction, Off-line event streaming, Physics analysis, Detector simulation

4. Value Delivered

***A* success or failure status code**

5. Participating Actors

No

6. Pre-Conditions

6.1 A collection of events is already exist and accessible by the actor.

6.2 The *Atlas Common Event Data Store* has remaining capacity and and accessible by the actor.

6.3 Information of the event collection is available from the *Atlas Bookkeeping System*.

Use case specification: Generalization of LAr-UC-1 (b)

7. Flow of Events

7.1 Basic Flow

The actor gets from the Atlas Bookkeeping System the list of event collections that have been recorded and which have not yet been fed into the Atlas Common Event Store.

If a query to the Atlas Bookkeeping System fails, a comprehensive message will be addressed to the actor.

The actor starts to feed a new event collection into the Atlas Common Event Store. This can be done partially in a run, or totally at once depending on the resource availability.

If the transfer succeeds, then the actor update the Atlas Bookkeeping System to keep track of successful operations and will be further used to automatically reach the event collection in the common software framework.

If a new event collection is not partially/totally readable, the Atlas Bookkeeping System should be appropriately updated to keep track of this problematic event collection.

If the transfer of an event collection into the Atlas Common Event Store fails, a clear diagnostic message must be provided in a log file and the Atlas Bookkeeping System must no be updated.

Use case specification: Generalization of LAr-UC-1 (c)

7.2 Alternative Flows

No alternative flows.

Use case specification: Generalization of LAr-UC-1 (d)

8. Invariant Conditions

8.1 The content of an event collection has to be invariant.

9. Post-Conditions

9.1 The Atlas Common Data Store must be closed properly.

9.2 The Atlas Bookkeeping System must be closed properly.

10. Special Requirements

No special requirements.

11. Remarks

11.1 There should be a person in charge to store an event collection to the Atlas Common Data Store.

11.2 Each event collection must have an unambiguous run number.

11.3 Run numbers are the primary identifiers of event collections and records in Atlas Bookkeeping System. Run numbers must then be unique.

Next steps

- **The template and an example usage of use-case specification is available.**
- **To start bottom up and top down approaches by interacting with subsystem software people.**