Using a Java Embedded Domain-Specific Language for LHC Test Analysis
(CERN, Geneva, Switzerland)

The main requirements for the Analysis framework are:

- The ability to define conditional checks called Assertions. These assertions have to ensure that a function of time is comparable to a value at a single moment or to another function of time over a time interval.
- The use of a graphical user interface (GUI) component to display the signals used in the assertions and the results of those assertions.

Requirements

Analysis Script

```java
public class MyModule extends AnalysisModule{
    asserting assertsThat(1 MEAS)
        .isLessThan(55.0, AMPERE) .at(PR_EVENT_TRIGGER);
    asserting assertsThat(1 MEAS)
        .isEqualTo(EXPONENTIAL_FUNCTION) .withinAt(2.0, AMPERE)
        .startingAt(30, MILLI(SECOND)) .after(PR_EVENT_TRIGGER)
        .endingAt(2, MINUTE) .after(PR_EVENT_TRIGGER);
}
```

Language Processing Steps

- Eclipse IDE
  - Analysis script
  - Java compilation
  - Valid script

- Analysis framework
  - eDSL compilation
  - Analysis Model
  - Execution
  - Analysis Result

Semantic Model

- Analysis
  - 1:N
- Assertion
- 1
- Condition (IsLessThan)
- 1
- Scalar Value (Signal Value at Point in Time)
- 1
- Constant Scalar (55.0 AMPERE)
- 1
- Signal (L_MEAS)
- 1
- Point in Time (PR_EVENT_TRIGGER)

Graphical User Interface

- A Time range is the period between two included points in time.
- A Point in time is a marker of a given moment in time.
- A Signal is a discrete (non-continuous) function with the dimension of time as X values and any physical dimension as Y values.

Summary and Outlook

The analysis framework will be ready to perform the operational analysis for at least one prototype test during the next hardware commissioning campaign in 2014. Further analysis modules will need some more features, which are in the pipeline for 2015.

Future plans for the analysis language concern the implementation of calculation functionalities for more general analysis of operational data of the CERN accelerators. The analysis language could also be used directly in other operational software. Examples of this are: Post Mortem Modules, Sequencer Tasks or Pre/Post Operational checks of systems.

For the Fluent API of our analysis language, we chose the following concepts:

- **Method chaining** consists of having the current object modifiers returning the host object carrying the next element of the language syntax. Using a set of Progressive interfaces, one can enforce the syntax tree so the user can only use the mandatory elements provided all along the chain.
- **Expression builders** are used to decouple the fluent API building logic from the semantic model execution logic.
- The **Semantic model** can be seen as immutable domain objects populated by the language using the expression builders.
- **Object scoping** provides a template for the writing of the user analysis script. In the Java language, one can use the Instance initializer technique to achieve the template implementation.

eDSL Concepts

Method chaining consists of having the current object modifiers returning the host object carrying the next element of the language syntax. Using a set of Progressive interfaces, one can enforce the syntax tree so the user can only use the mandatory elements provided all along the chain.

Expression builders are used to decouple the fluent API building logic from the semantic model execution logic.

The Semantic model can be seen as immutable domain objects populated by the language using the expression builders.

Object scoping provides a template for the writing of the user analysis script. In the Java language, one can use the Instance initializer technique to achieve the template implementation.