GAUDI Scenarios

Great Architecture for Unified Data Interaction

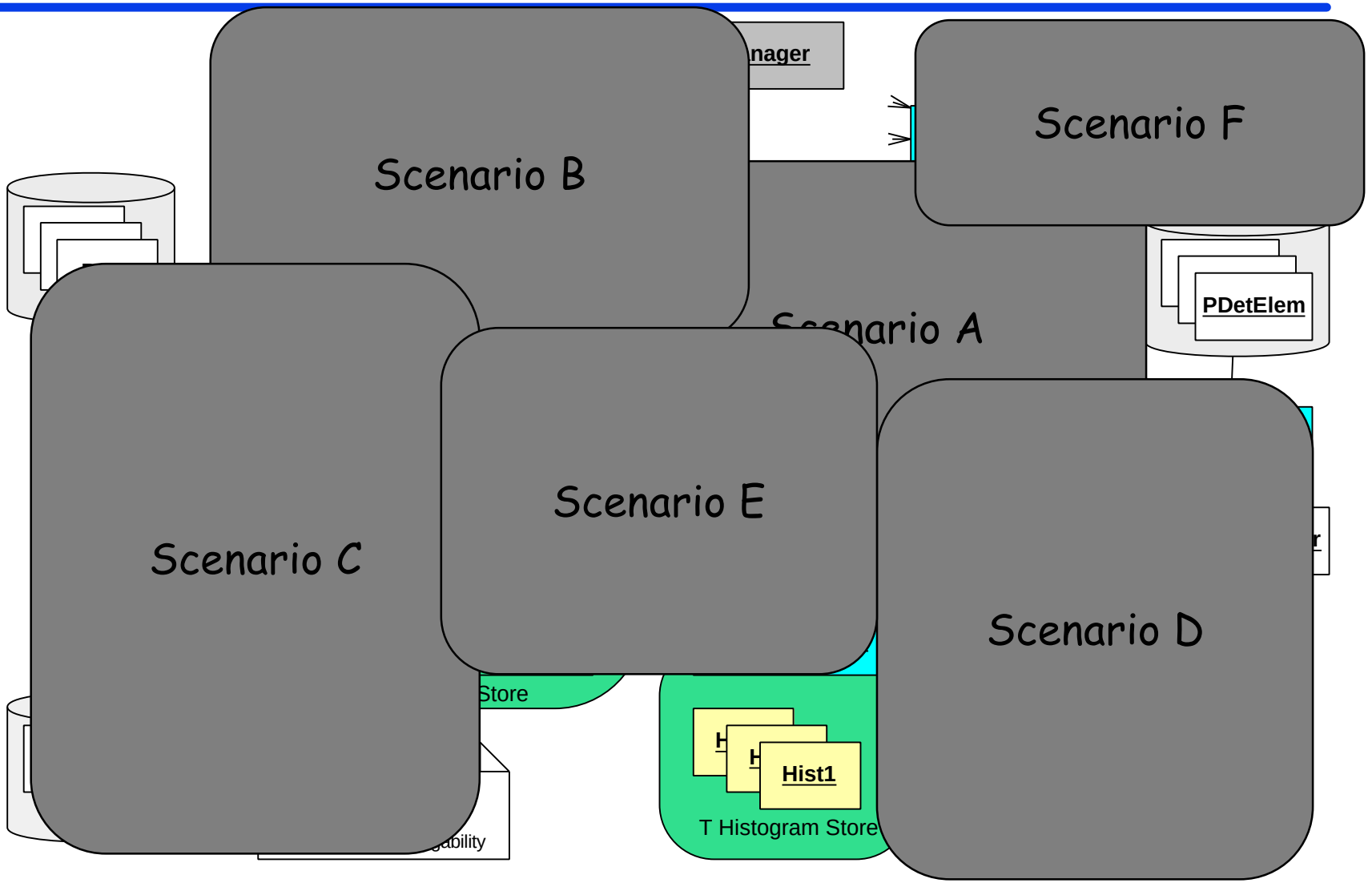
10 November 1998

P. Mato, CERN

Scenarios: Why?

- A *scenario* is a brief description of a single interaction of a *stakeholder* with a system (concept similar to *use case* from the object-oriented community).
- There is a scenario-based method for analyzing architectures (SAAM).
- They provide a means to characterize how well an architecture responds to the demands placed on it.
- We need the aid of *stakeholders* for creating and organizing scenarios:
  - Physicists doing analysis, sub-detector software developers, people responsible of productions, framework developers,...

Scenarios: Coverage



Scenarios: Classification

- We divide the scenarios into four categories according to the type of user:
  - (A) Scenarios which deal with the **use** of applications built within the framework. **Functionality** of the system.
  - (B) Scenarios which deal with the **development** of components built within the framework.
  - (C) Scenarios which deal with **configuration** management.
  - (D) Scenarios which deal with the interaction of the framework with the **environment** and handling of the **change**.

Examples of type (A)

- An analysis job runs several algorithms to select events of a run and produces a set of histograms which are saved at the end of the job. ➔ *basic functionality for release 1.0*
- A reconstruction job is run over several runs. The detector alignment constants change between two of the runs used. ➔ *automatic synchronization of the detector data and the current event data.*
- A track is displayed. A physicist wishes to select a subset of the track's hits (by clicking on the event display) and refit the track using only those hits. The new track is to be displayed along with the original. ➔ *user full interactivity for analysis and debugging.*
- A detector calibration type job is run: several algorithms are called once per event to collect statistics, but after a given number of events another set of algorithms are run once to calculate new calibration constants. ➔ *calibration type of job*

Examples of type (B)

- There is a central database of generated data. Two independent reconstruction developers read this data and produce their own data types. Both wish to save their objects along with references to the objects in the original database. ➔ *independent development.*
- A user invents a new data type (e.g. a kinked-track) and wishes to store it to disk. What are the necessary steps needed ➔ *no need for centralized management.*
- A physicist wishes to define a new graphical representation of a reconstructed object ➔ *the system is extendable*
- A algorithm makes use of other more basic algorithms. In case of a fatal numerical exception, the culprit event must skipped and the job continued ➔ *error handling.*
- The units of a measurement has changed from a given run. The algorithm must cope in a transparent way the runs before and after the change. ➔ *data versioning.*

Examples of type (C)

- Monte Carlo data is produced with a new version of the detector geometry. A label is associated to the detector description version and the produced data. ➔ *labeling detector geometry versions.*
- The data is reprocessed with new calibration and alignment constants. After the new data is checked against the old one, the old reconstructed objects are deleted from the database and the new ones made the default. ➔ *data management.*
- The new test beam apparatus setup is introduced. The new collected data is going to be analyzed using the new setup while the previous test beam data is still analyzable. ➔ *concurrent versions of test beam setups.*

Examples of type (D)

- A set of basic reconstruction algorithms must be run in the DAQ system as part of level-2 or level-3 trigger algorithms. ➔ *applications may run on the online as well as offline environments.*
- The operating system of a supported platform changes. ➔ *operating system independence.*
- It is necessary to replace the database system used for event data storage due to the lack of performance. ➔ *ODBMS independence.*
- We decide to change programming language, e.g. to Java. ➔ *Language independence*

What next?

- We need to collect as much as possible scenarios (Paul's mail). Interview stakeholders.
- How many we need?
- For the architecture review, we will select something like a dozen scenarios that cover most of the functionality and qualities of the system.
- The selected scenarios will be analyzed in detail.
  - Functionality type (A) can be illustrated using "interaction diagrams"
  - Qualities scenarios will be evaluated.