Exceptional service in the national interest

SAND2014-4497P

Processing Architecture Concepts for IDC Reengineering

Mark Harris

June 2-4, 2014

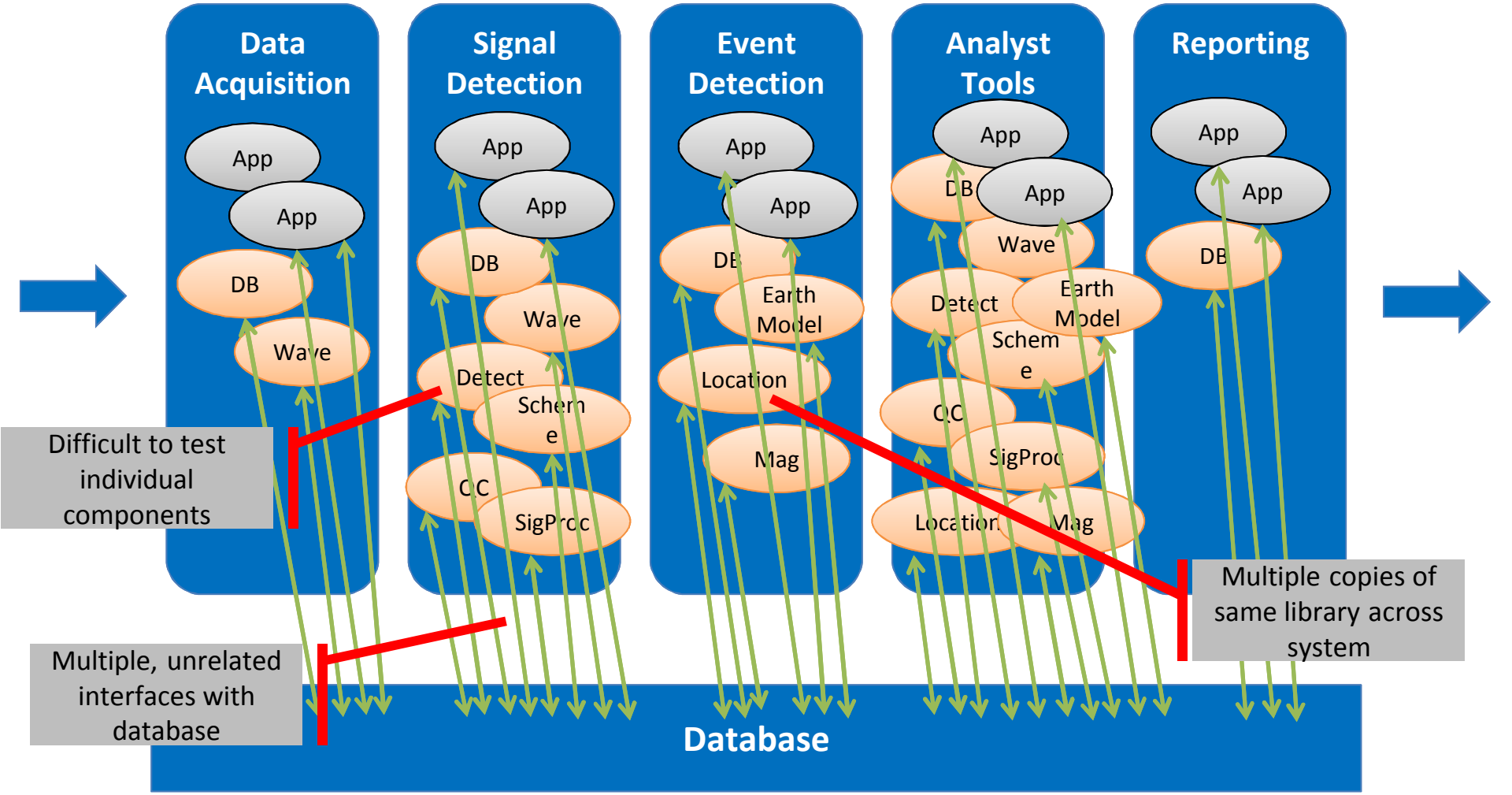
Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation a wholly owned subsidiary of Lockheed Martin Corporation for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Overview

Sandia National Laboratories

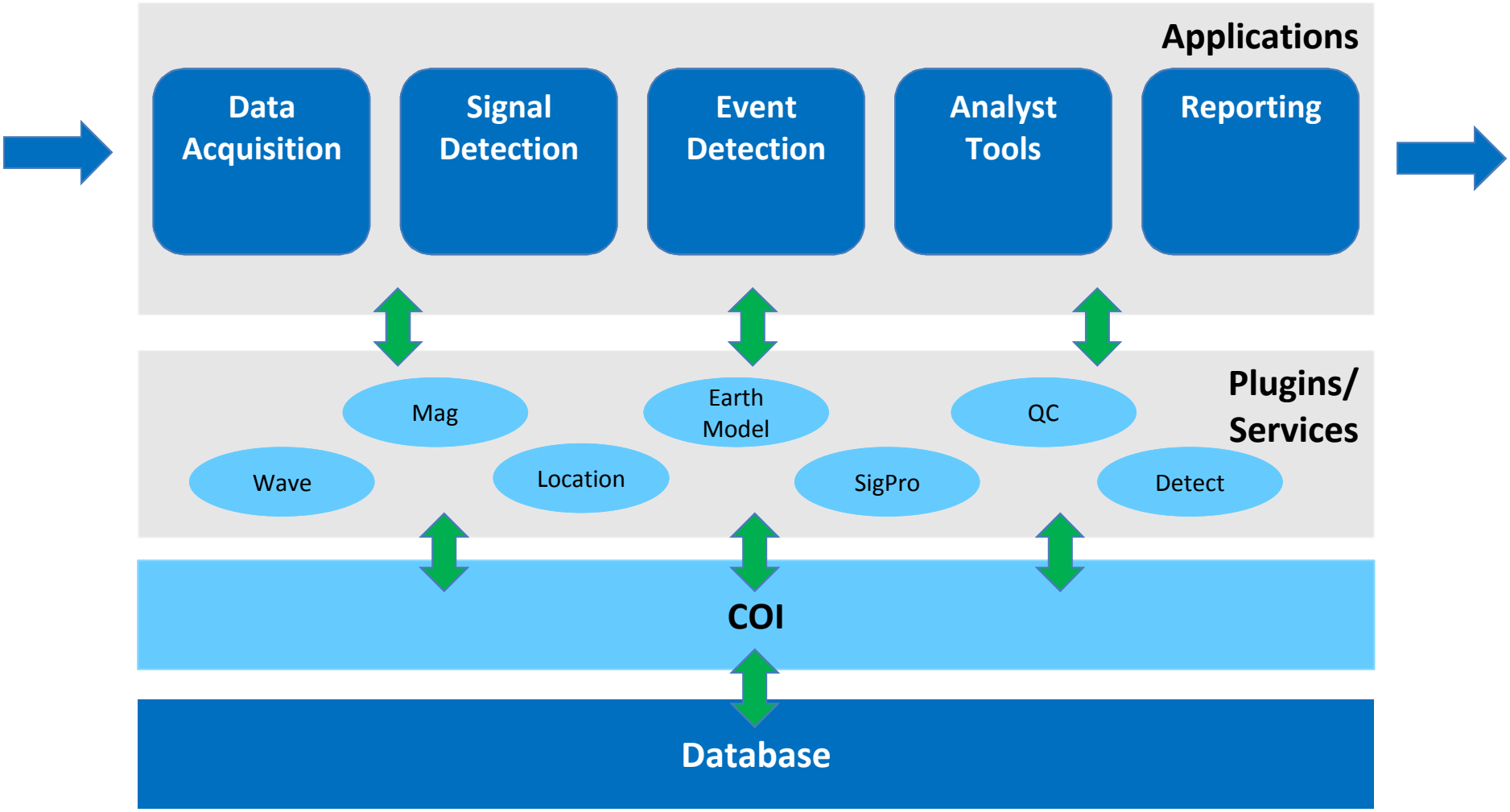
- The Next Generation IDC system should support
  - Easy integration of new algorithms and geophysical models
  - Dynamic selection of algorithms
  - Configurable processing sequences

Current Architecture



Future Architecture

Sandia National Laboratories



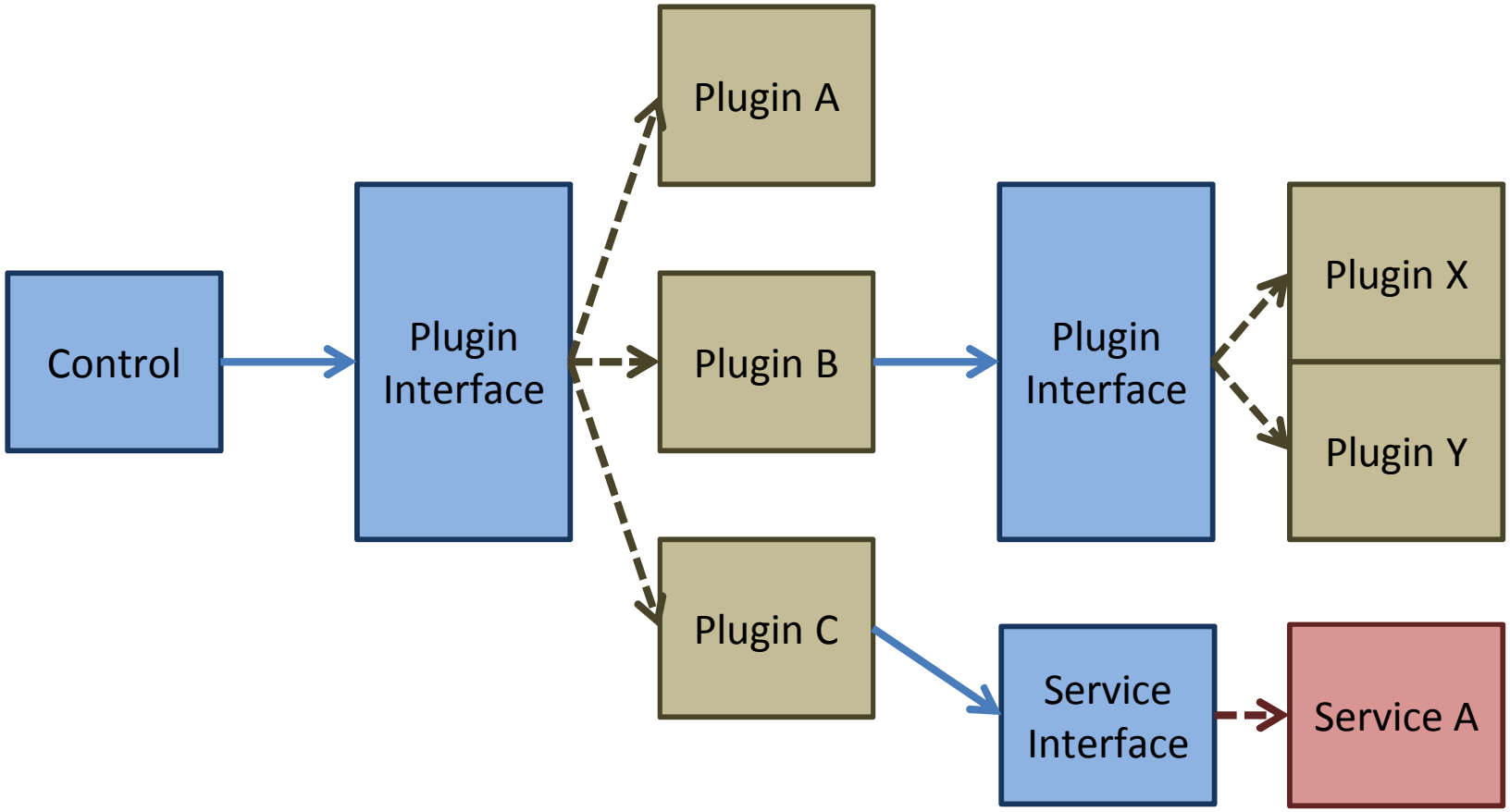
Architecture to Support New Algorithm / Model Integration

- Key for maintaining system is updating algorithms
  - Location
  - Signal Detection
  - Association
  - Etc.
- Facilitate model updates independent of algorithm updates
  - Earth models, etc.

Plug-in Architecture

- Define algorithm interfaces
- Decouple the system from the algorithms through abstract interfaces
- Support multiple algorithm implementations in the system
- Determine system to algorithm coupling dynamically at run time

Processing Plugins and Services



Sequencing Processing

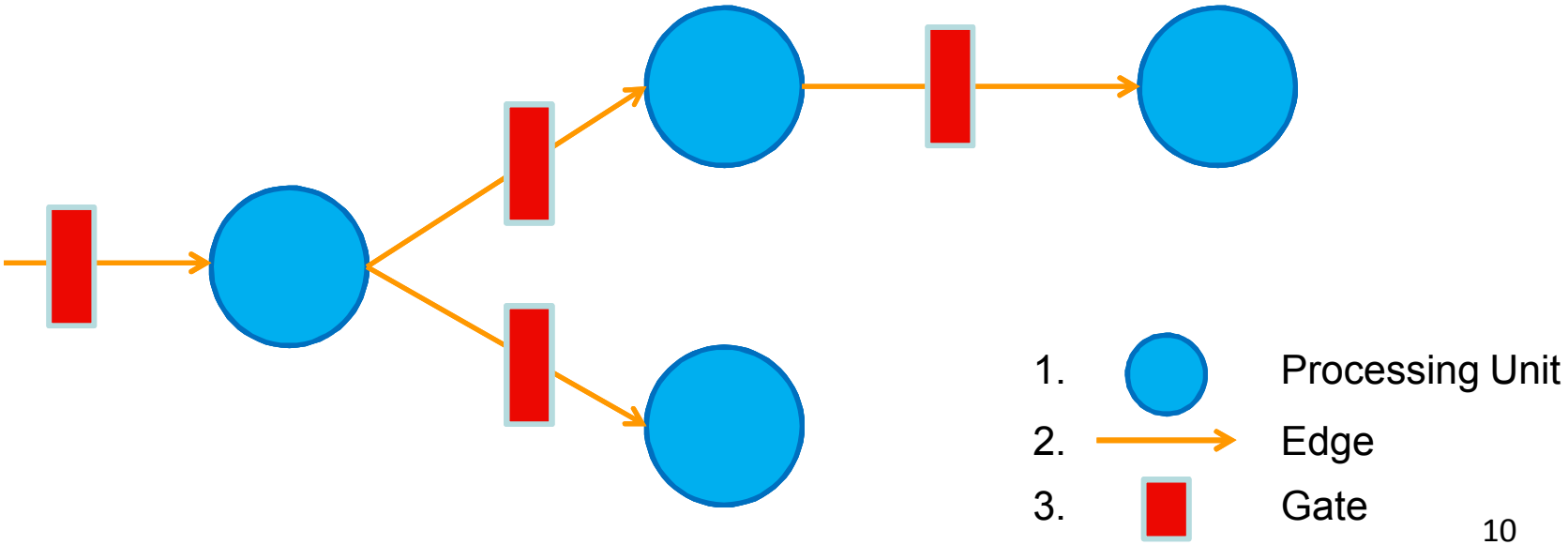
- Decouple the processing sequence (order in which algorithms are applied) from the code that implements the algorithms
- Provide an interface to define the order
- Implement a framework to call algorithms in the defined order
- Implement algorithms without dependencies on order
  - Error checking if preconditions are not met

Key concept: Process Sequence

- A *Processing Sequence* consists of:
  - 1. Which processing operations to run (e.g. signal detection, location, …)
  - 2. The relative order for running those processing operations
  - 3. Rules describing when processing is allowed to run

Key concept: Processing Sequence

- Processing Sequences organize these components as a data flow graph
  - 1. Which processing operations to run (*Processing Units*)
  - 2. Relative order in which *Processing Units* run (defined by *Edges*)
  - 3. Rules describing when *Processing Units* are allowed to run (*Gates*)



Processing Unit

- *Processing Units* invoke <<control>> classes, e.g.
  - *Data Quality Control*
  - *Signal Enhancement Control*
  - *Signal Detection Control*
  - *Signal Detection Event Builder Control*
  - *Event Location Control*
  - etc.

Edge

- An *Edge* represents a data flow between two *Processing Units*
- A *Processing Unit* may have multiple incoming *Edges*
  - Data on incoming *Edges* is sent to the <<control>> classes for processing
- A *Processing Unit* may have multiple outgoing *Edges*
  - Results from invoking <<control>> classes flow onto outgoing *Edges*

Gate

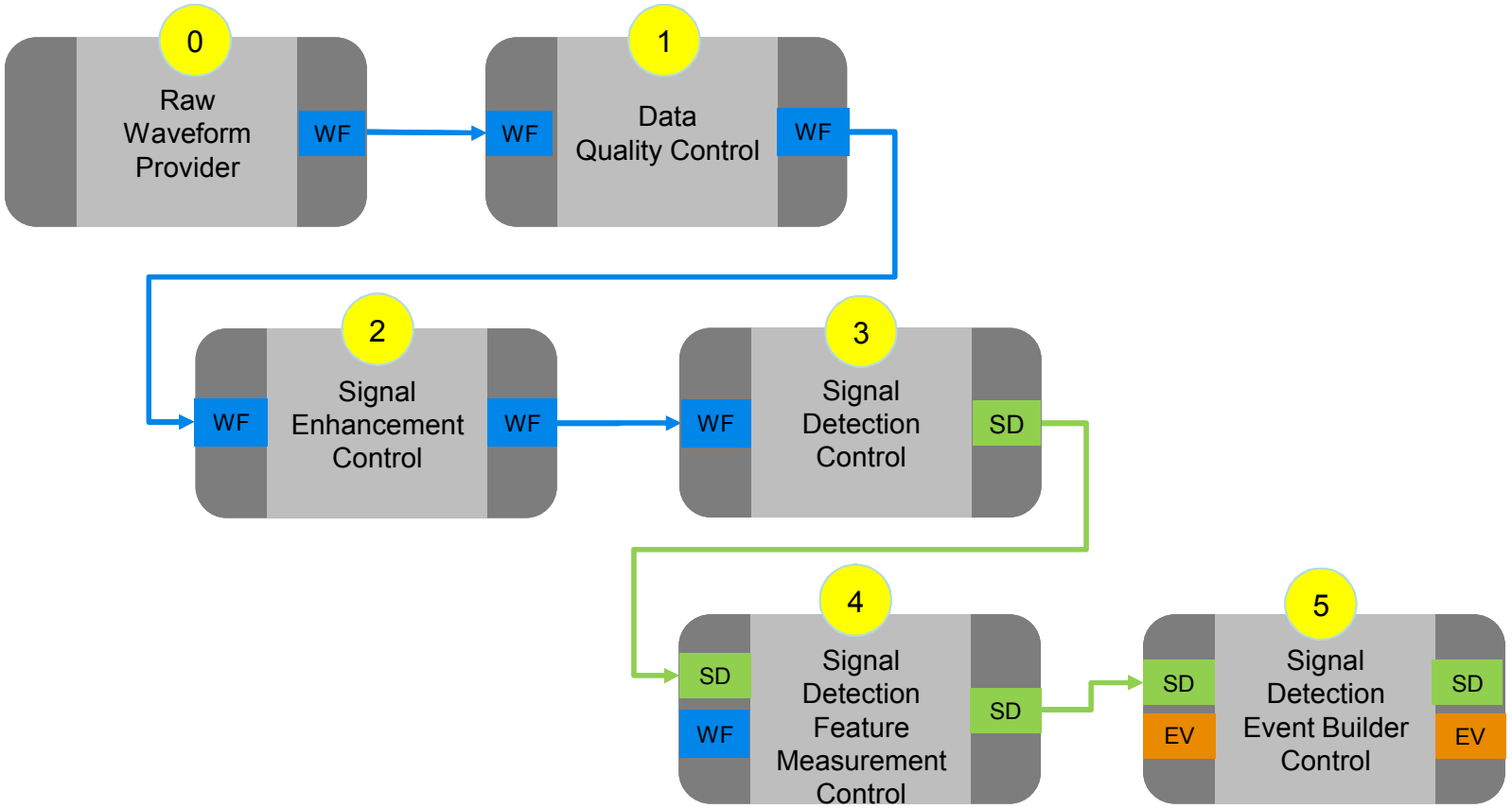
- *Gates* guard access to one or more *Processing Units*
- When a *Gate* opens all of the *Processing Units* it guards are allowed to run

- Each *Gate* contains one or more *Gate Conditions*
  - A *Gate* opens when all of its *Gate Conditions* are satisfied
- There are currently two types of *Gate Condition*
  - *Data Available Gate Condition* – satisfied when there is data flowing on the *Edge* associated with the condition
  - *Timed Gate Condition* – satisfied when a certain amount of time has elapsed since the condition was previously satisfied
  - New *Gate Condition* types will likely be added as the UCRs are developed

Key Concept: Processing Sequence Control

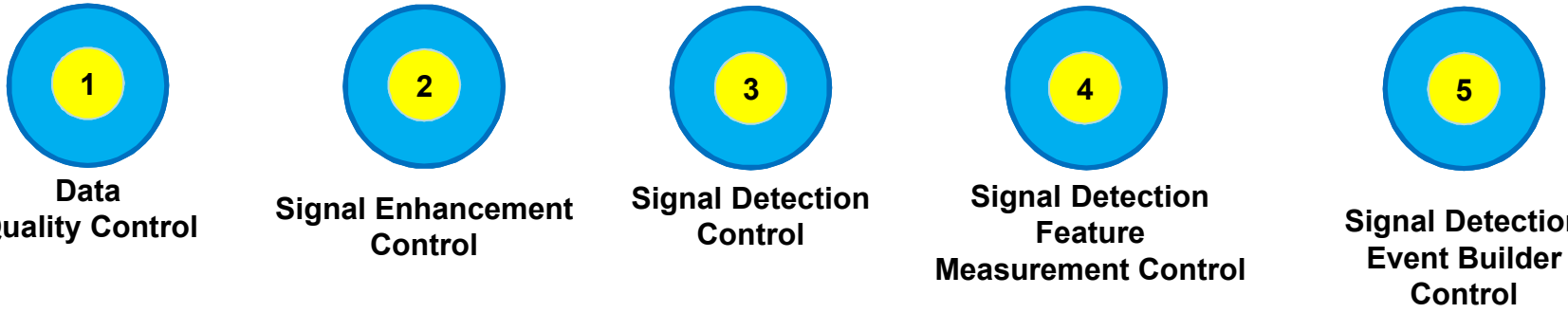
- Envision a *Processing Sequence Control* that executes *Processing Sequences*
- Regularly stimulated by the *System Clock* to:
  - Evaluate *Gates* to find which *Processing Units* are ready to execute
  - Execute the ready *Processing Units*

Example 1: Traditional Pipeline (partial)



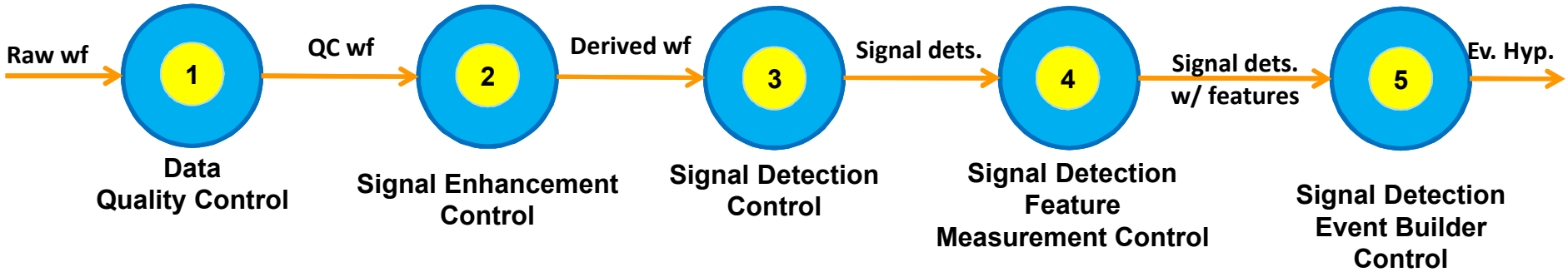
Example 1: Traditional Pipeline (partial)

1. Define the *Processing Units.*



Example 1: Traditional Pipeline (partial)

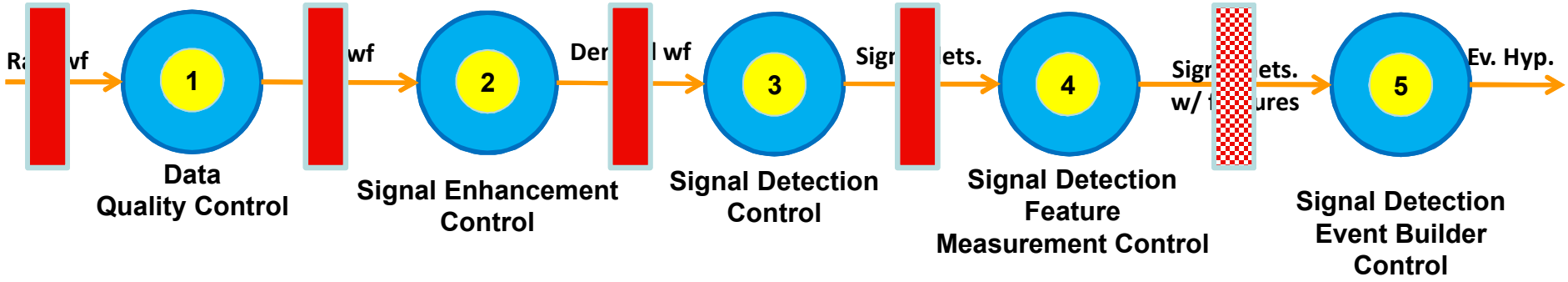
2. Define *Edges*

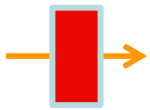


Note: data types are included on these Edges to show what type of data flows on each Edge. Edges do not have types.

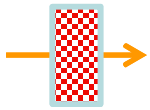
Example 1: Traditional Pipeline (partial)

3. Add *Gates* and their *Gate Conditions*





Gate opens if:
Data Available* on covered *Edge



Gate opens if:
Data Available on* covered *Edge
AND
Timed Gate Condition (x minutes)

Note: data types are included on these Edges to show what type of data flows on each Edge. Edges do not have types.

Backup

Sandia National Laboratories

Goals for a next generation system

1. Enhance existing mission capabilities
2. Re-architect software using modern practices
3. Support incremental improvement
4. Develop/integrate state-of-the-art algorithms
5. Migrate to generic hardware
6. Integrate improved geophysical models
7. Test to ensure success
8. Address new requirements

Envisioned Improvements

Sandia National Laboratories

- Establish modern software engineering process based on the Rational Unified Process (RUP)
- Develop Model-Based software architecture
- Extensible – Service Oriented Architecture
- Streamline R&D integration capability
- Improve analyst tools & workflow – e.g. undo/redo
- Event communications – social technology
- Comprehensive geophysical model support
- Geographic processing configuration model
- Updated data model & common object interface
- Capture and use processing history
- Integrated system and mission test capability
- Open platforms to reduce vendor lock (hardware, operating system, database)

Focus on Architecture

- In software development, we aim at getting the architecture designed, implemented, and tested early in the project. This means that, early in the project, we focus on the following goals:
- Defining the high-level building blocks and the most important components, their responsibilities, and their interfaces.
- Designing and implementing the architectural mechanisms, that is, ready-made solutions to common problems, such as how to deal with persistency or garbage collection.
- By getting the architecture right early-on, we provide a skeleton structure for our system, making it easier to manage complexity as we add more people, components, capabilities, and code to the project. We also identify what reusable assets we can leverage, and what aspects of the system needs to be custom built.

What is Architecture?

"The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships between them." *Wikipedia, the free encyclopedia*

"The fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution." *ANSI/IEEE*

"Software architecture is the set of design decisions which, if made incorrectly, may cause your project to be cancelled." *Eoin Woods*

What is Architecture?

- Design the system from a high level
- Capture the design in a formal manner
- Communicate the logical design of the system to people who care (stakeholders)