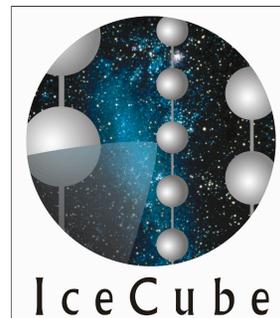
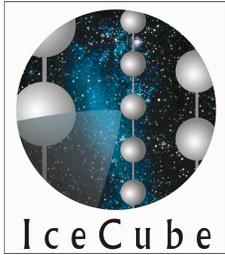


Surface-DAQ Event Detection System (Triggering) Status

Dan Wharton (U. Wisc.)



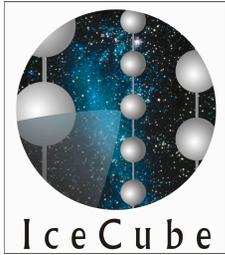
**IceCube Collaboration Meeting
March 20, 2005**



Surface-DAQ Event Detection System

DAQ System status

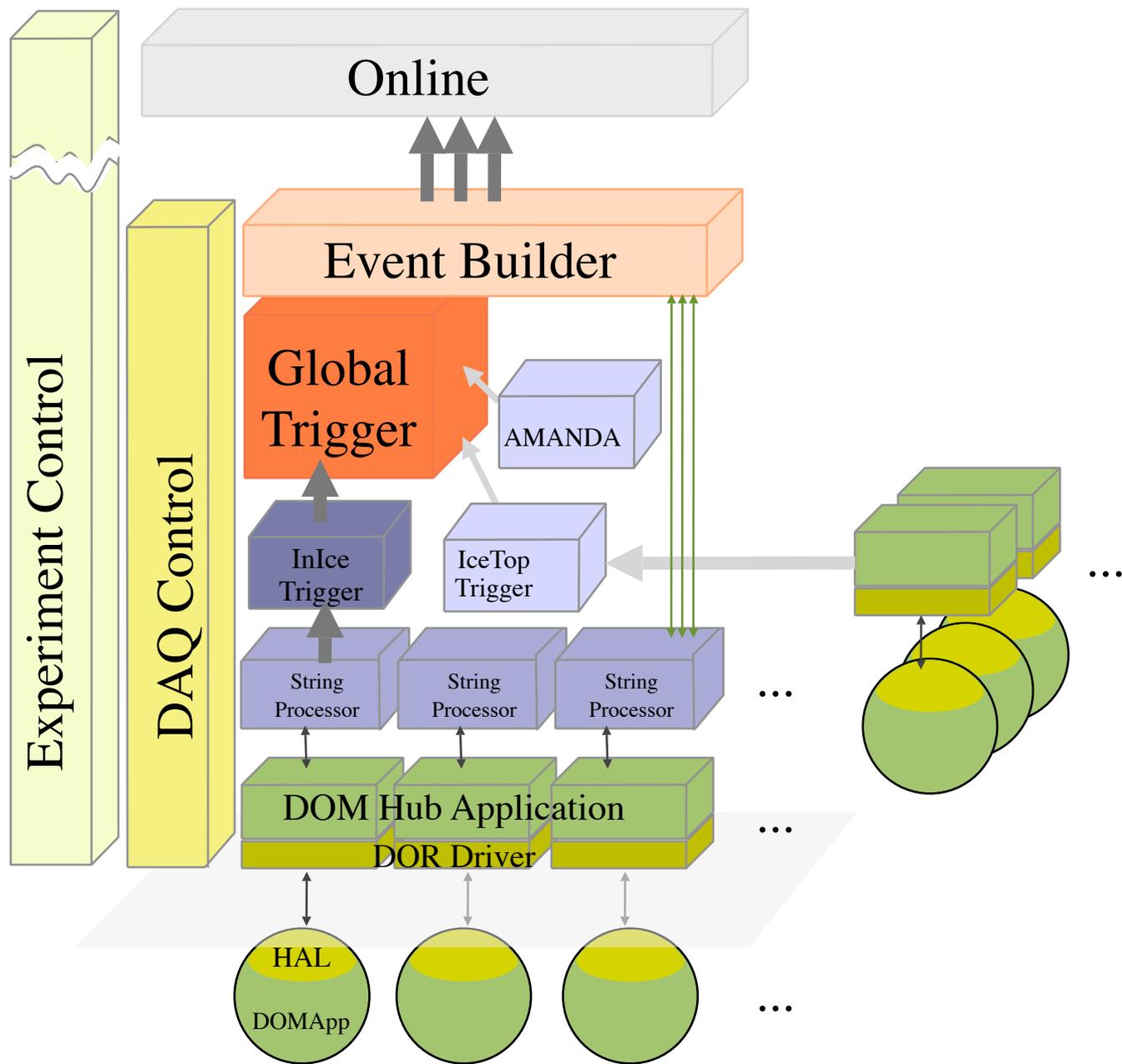
- Where did we start ?
 - Component Requirements
 - Interface Definitions
- What have we accomplished ?
 - Payload Framework
 - Trigger Algorithm Development
 - DAQ Component Development
 - DAQ-Monolith version 01-00-00 at the South Pole
- Where are we going?
 - Milestones of development



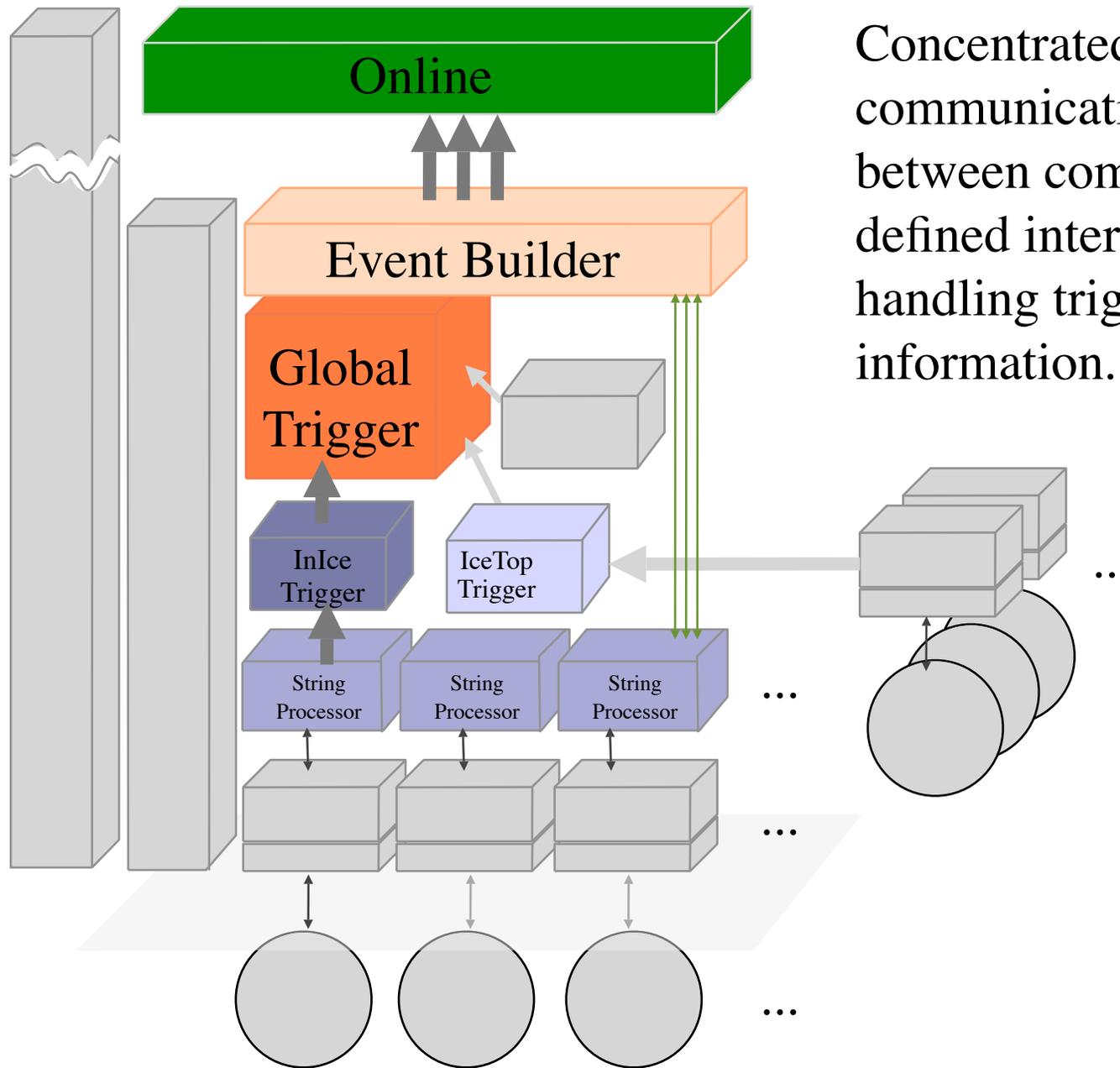
Where did we start ?

**DAQ System
status**

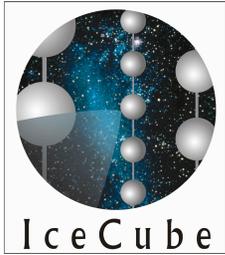
- Component Requirements
 - String Processor, Payload Framework (Dan Wharton, Keary Cavin UW)
 - IceTop DataHandler (Divya Swarnkar, BARTOL)
 - InIce Trigger (Pat Toale, PSU)
 - IceTop Trigger (Dave Seckel, Divya Swarnkar, BARTOL)
 - Global Trigger (Seon-Hee Seo, PSU)
 - Event Builder (Marc Hellwig, Mainz)



IceCube DAQ Architecture



Concentrated on communications protocol between components and defined interfaces for handling trigger and hit information.

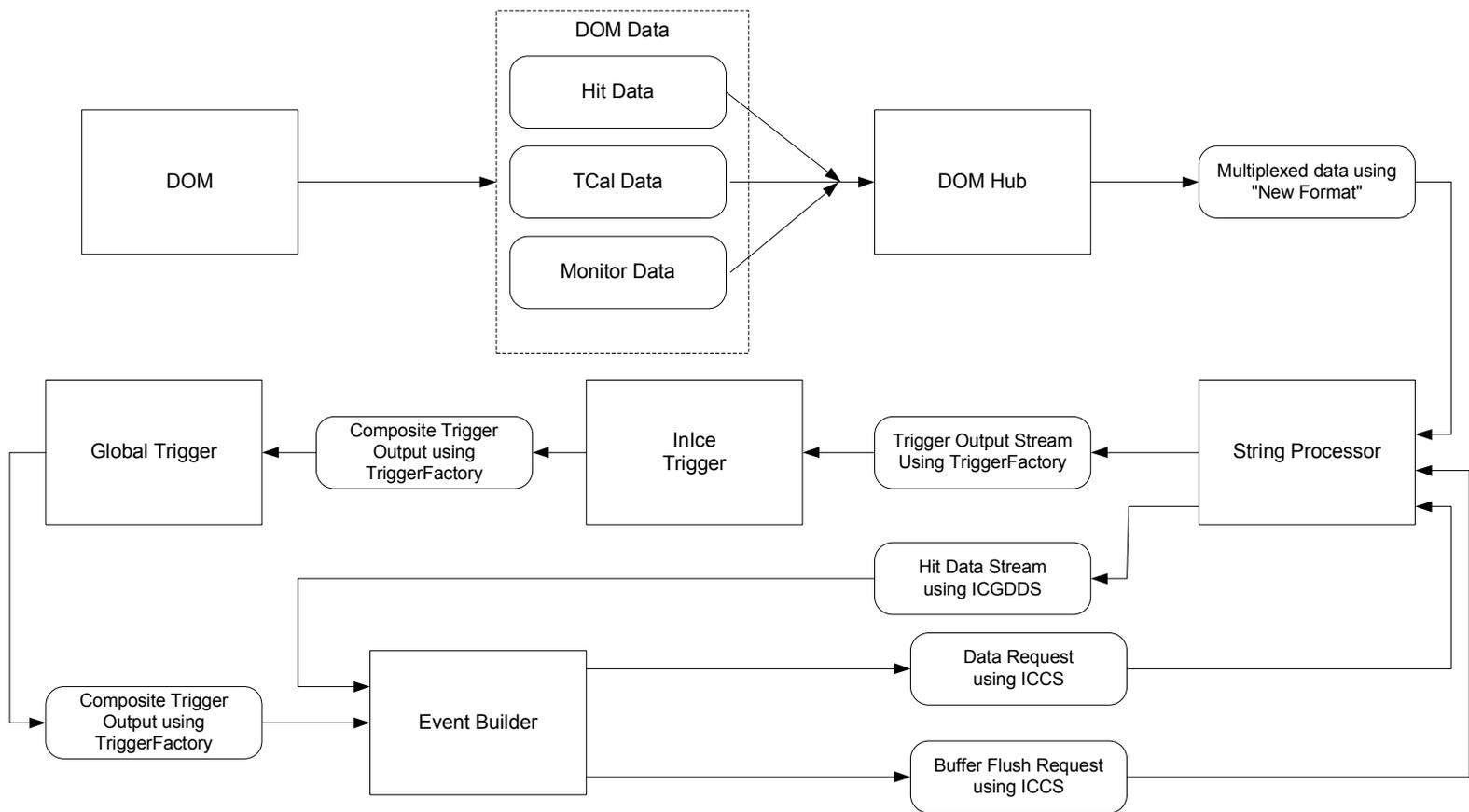


Interface Definitions

*DAQ System
status*

- **Fall Meeting at PSU**
 - Defined Information hierarchy needed for each component in the chain.
 - Created Payload interfaces which allowed work to proceed in parallel while programming to interfaces.
 - Defined system to be easily integrated to Splicer for each component.
 - Defined methods to read and write binary data, keeping 'on-the-wire' formats centrally located.
 - Defined framework for containing Trigger and Event data.

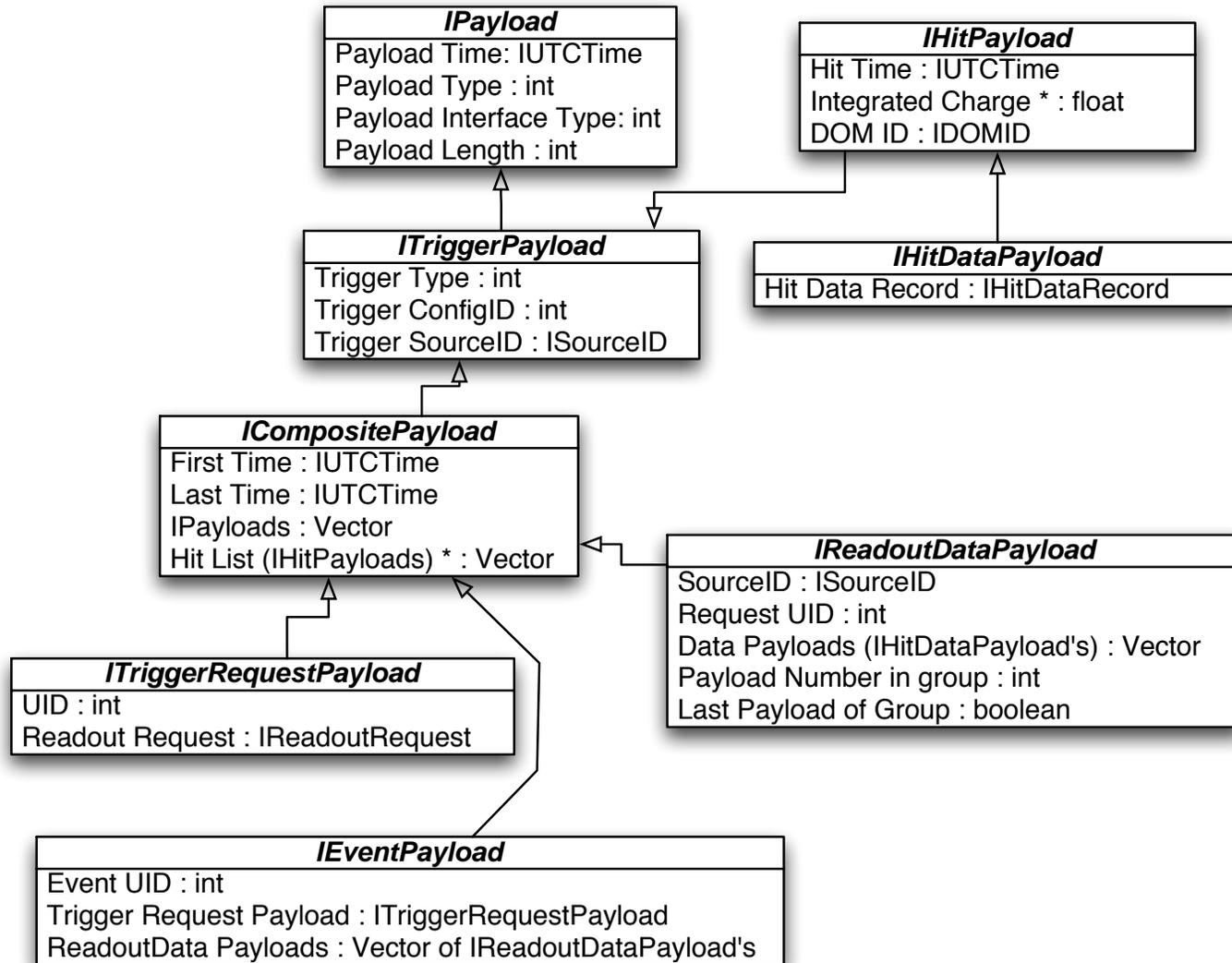
Defined Component Communications Requirements

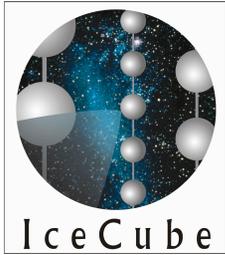


DAQ Payload Framework Interfaces

version 01-00-00

* = not implemented yet

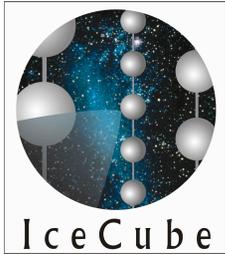




Surface-DAQ Event Detection System

DAQ System status

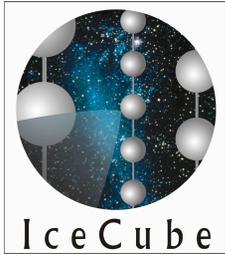
- Where did we start ?
 - Component Requirements
 - Interface Definitions
- **What have we accomplished ?**
 - Payload Framework
 - Trigger Algorithm Development
 - DAQ Component Development
 - DAQ-Monolith version 01-00-00 at the South Pole
- Where are we going?
 - Milestones of development



Interface Implementation

*DAQ System
status*

- Payload Framework constructed as single source of object implementation.
 - Simplified debugging and testing
 - Single source code base for binary 'on-the-wire' formats
 - Allowed work to go on in parallel letting Trigger and Event Builder code to be built on common code base, reducing scope.



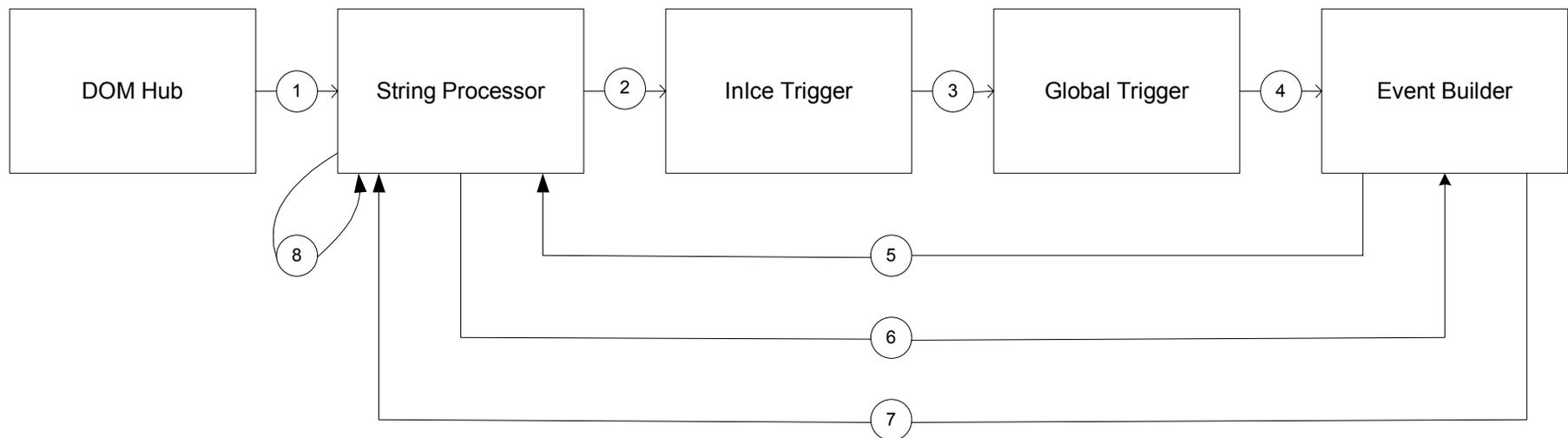
Interface Implementation

*DAQ System
status*

- Abstracted I/O for Payloads
 - Allowed testing of components on an individual basis.
 - Allowed input of TestDAQ Hit data for input to test systems.
 - Allowed component black box testing.
 - Allowed components to be ‘chained’ together outside of Global DAQ Framework (unit testing).
 - Paved the way for Monolith while reducing extra coding requirements.

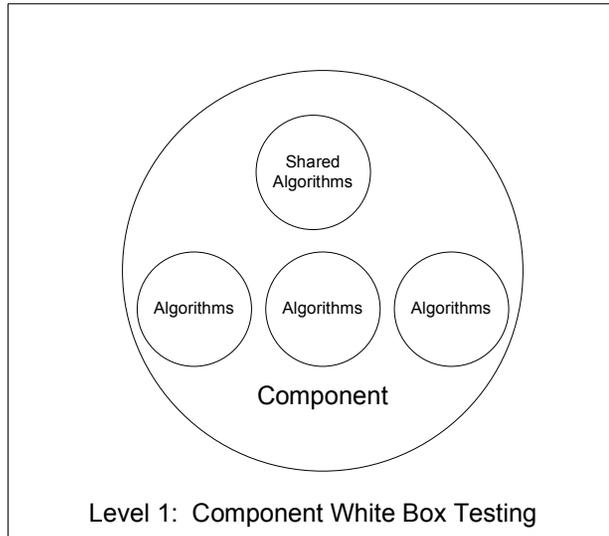
Used interfaces to simplify component development

Component Composition and Connectivity Testing



Defined Internal Component Testing Requirements

Testing Conducted by Component Developers

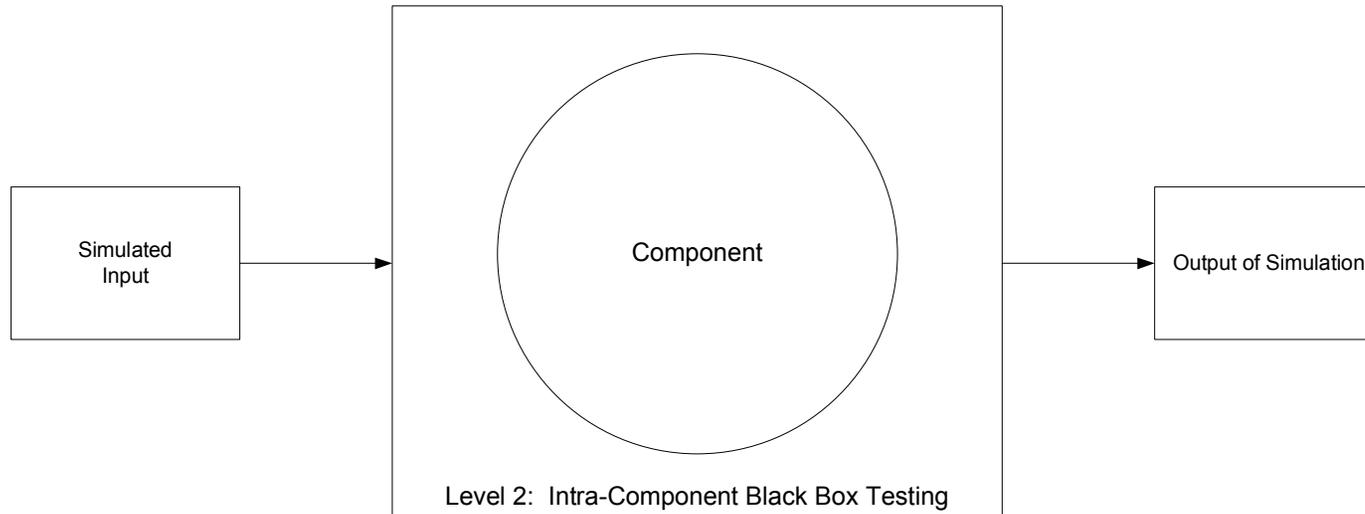


Each of the IceCube DAQ components (subsystems) will be subjected to white box testing, a software testing technique where explicit knowledge of the internal workings of the item being tested are used to select the test data. Each component developer is responsible for conducting white box testing.

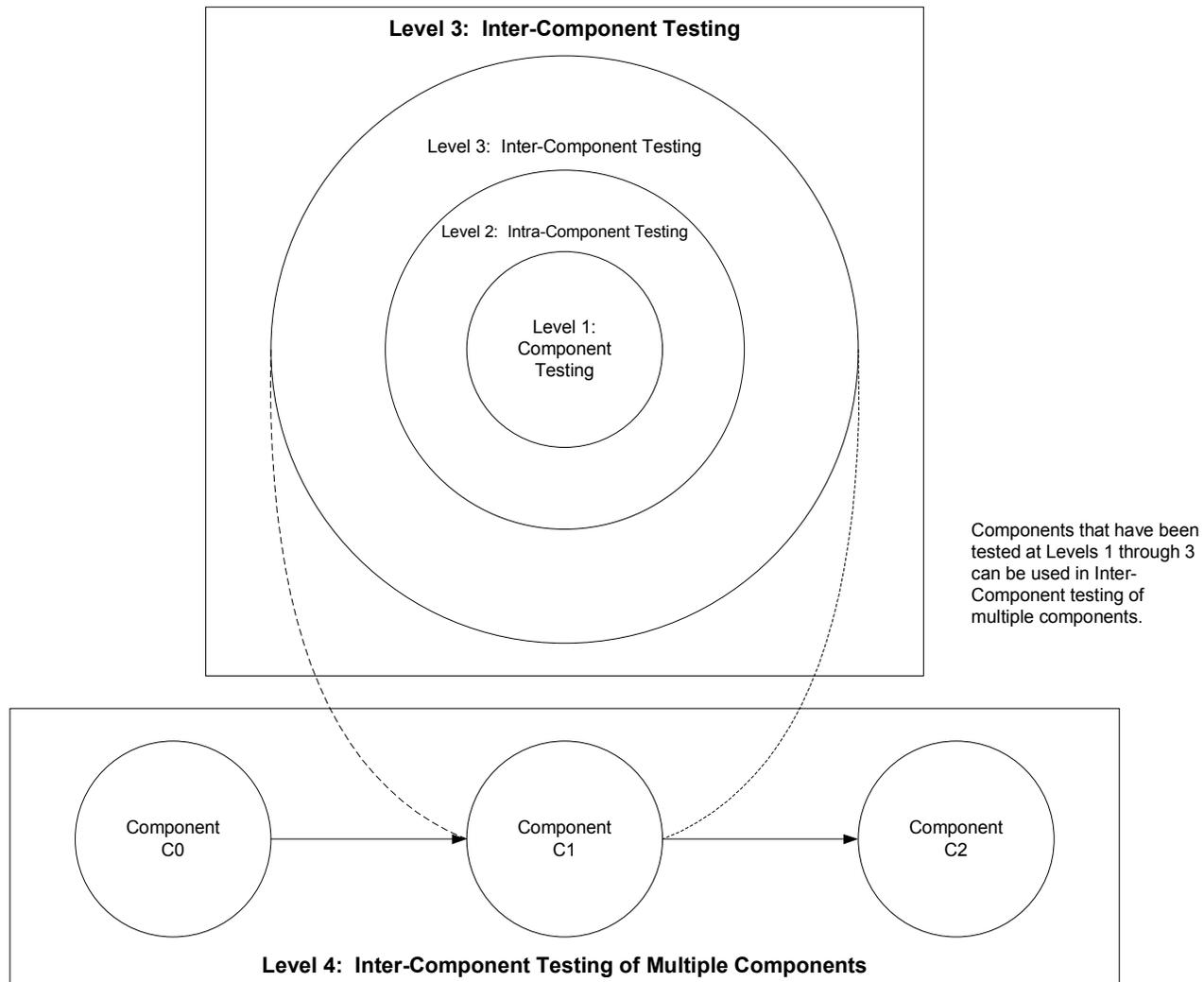
Each component consists of algorithms that are shared with other components and algorithms that are unique to the particular application.

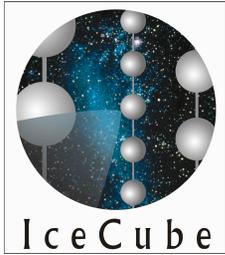
Black box testing (functional testing) will be conducted on each component in the system. In a black box test on a software design the tester only knows the inputs and what the expected outcomes should be and not how the program arrives at those outputs. The tester does not ever examine the programming code and does not need any further knowledge of the program other than its specifications.

Typical or expected input data is provided to the black box component and the outputs are inspected to determine in the black box component provides the proper results or functionality.



Defined 'Chained' component testing using flexible I/O

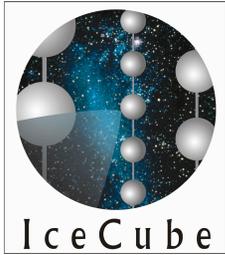




Surface-DAQ Event Detection System

DAQ System status

- Where did we start ?
 - Component Requirements
 - Interface Definitions
- What have we accomplished ?
 - **Payload Framework**
 - **Trigger Algorithm Development**
 - **DAQ Component Development**
 - DAQ-Monolith version 01-00-00 at the South Pole
- Where are we going?
 - Milestones of development

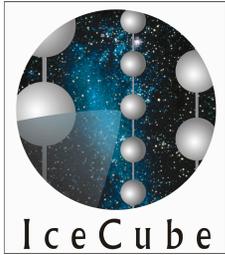


DAQ Component Development

*DAQ System
status*



- January '05 Integration Meeting at UW-PSL
 - Defined requirements for output of Event Builder and created libraries to decode event output as input to IceTray DataClasses.
 - Defined Milestones which serve both testing and instrumentation requirements.
 - **Monolith** - Single application (TestDAQ input)
 - **IO** - Multiple independent components (TestDAQ input)
 - **Discovery** - Global DAQ Framework hosting multiple independent components collecting data from DomHubs.

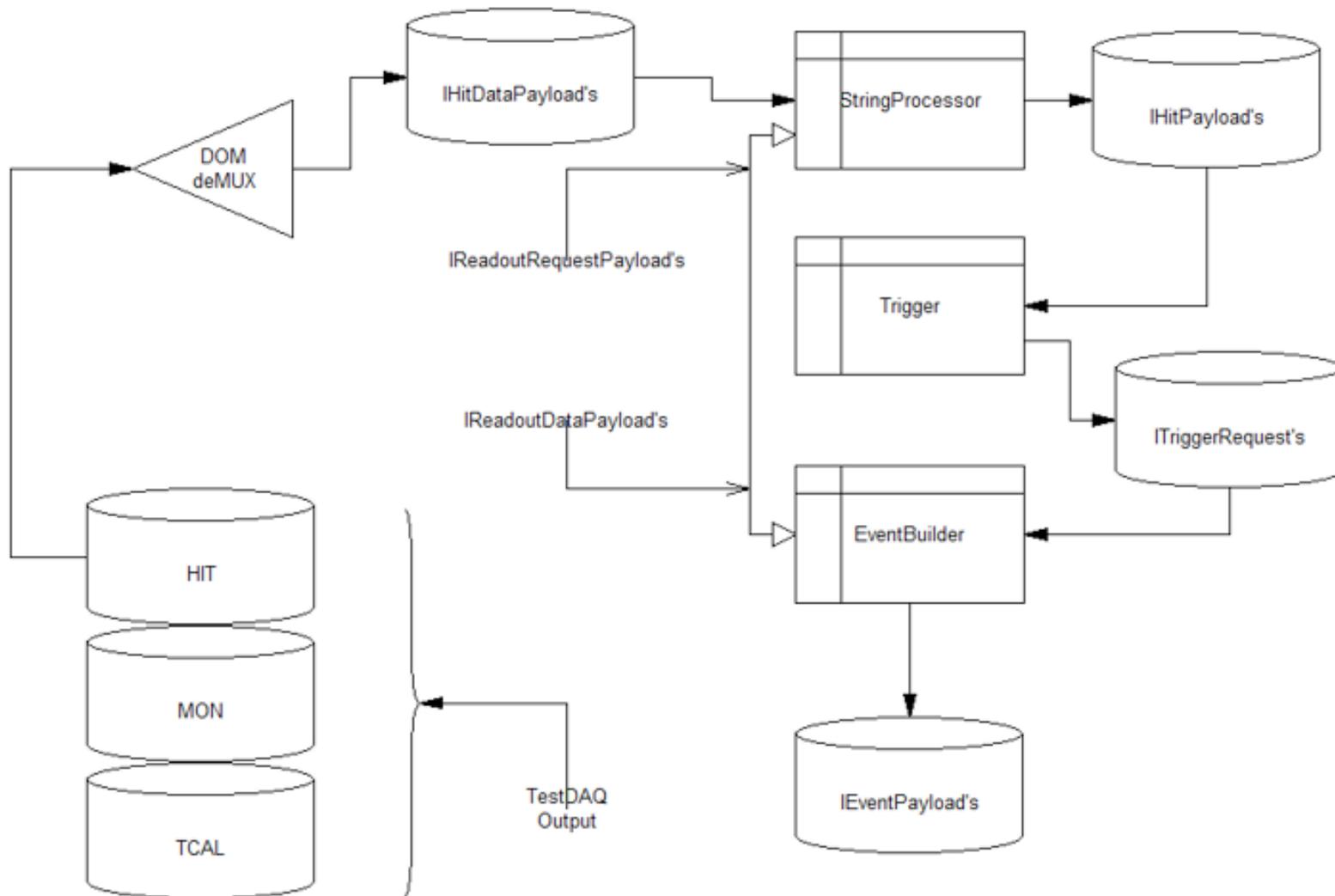


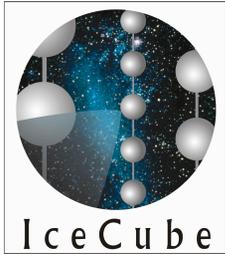
Monolith

DAQ System status

- Monolithic program which combines DAQ components outside of the DAQ Global Framework.
- Uses abstracted I/O in Payload Framework to input Hits from TestDAQ output.
- Tests components by 'chaining' input and output.
- Tests Payload Framework implementations.
- Produces output from Event Builder which is identical to output to be produced from DAQ Global Framework. This output is made available to Ice Tray through provided decoder library. Single point of access for event data as an interface reducing amount of replicated code to maintain. (Erik Blaufuss, Marc Hellwig)
- Provides vehicle for testing before next generation of DOMHub output is available.
- Adapted to be used at South Pole for producing events. Triggers produce good agreement with independent offline analysis.
- Limited life span, good testing platform but missing database and scalability advantages of the Global DAQ Framework.

Monolith Event Processor

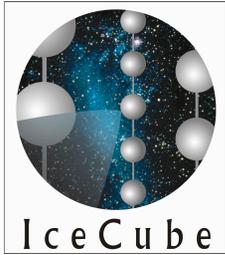




Surface-DAQ Event Detection System

DAQ System status

- Where did we start ?
 - Component Requirements
 - Interface Definitions
- What have we accomplished ?
 - Payload Framework
 - Trigger Algorithm Development
 - DAQ Component Development
 - DAQ-Monolith version 01-00-00 at the South Pole
- **Where are we going?**
 - Milestones of development

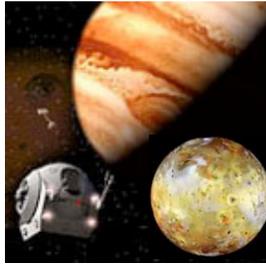
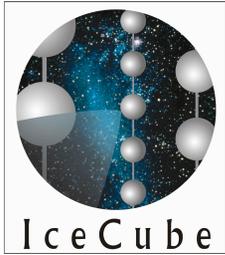


Monolith

*DAQ System
status*



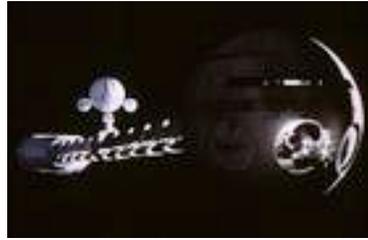
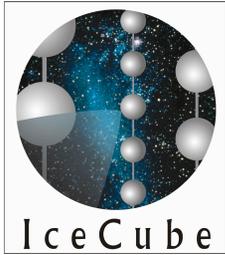
- At least one more release appropriate for the Pole before end of life.
- Additional Features
 - New fields in Event Payload
 - Combining IceTop and InIce triggers to a single pass.



IO

DAQ System status

- Multiple independent DAQ Components which are 'chained' together between separate processes instead of a single monolithic program.
- Progressive movement from file-based input/output to a functioning channel based processing system.
- Deployment of individual components into the DAQ Global Framework in which only the StringProcessor component and IceTop DataHandler has special considerations.
 - Data Input still mainly from TestDAQ for system development
- As development continues movement from TestDAQ to DOMHub data.
- Parallel development of StringProcessor and IceTopDataHandler for processing next generation DOMHub data.
 - High Speed Cache of Time ordered Hit data
 - Integral Time calibration from TCAL stream instead of using Rapcal from output of DataCollector. Verification of consistency of time calibration with with TestDAQ values when run in that mode.
- Data throughput analysis and performance testing
- Thoroughly test subsystems in preparation for 'Discovery'

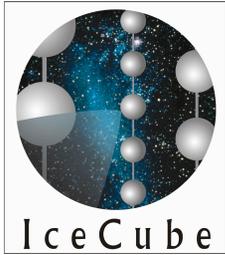


Discovery

DAQ System status



- Global DAQ Framework with all components deployed in JBOSS.
- All components configured through DAQ Control and configuration.
- Data Collection occurs directly from instrument through DOMHubs.
- Spiral Development Plan with functional milestones.
- Provide scalable DAQ for larger instrument array.



Conclusions

DAQ System status

- Designed and created necessary infrastructure to support Triggering and flexible staged development.
- Chose development path with measurable and useful milestones.
- Achieved successful deployment of Monolith milestone.
- Are building on pre-tested sub-components as DAQ increases in complexity which reduces risk and increases reliability.
- Are poised to move from Monolith to IO and on to Discovery.

