Outline description of an "Open Framework" for the Ptolemy based high level modelling initiative

Authors : R.Bock,P.Clarke,G.Crone, R.Hughes-Jones, K.Korcyl, W.Li, S.Wheeler,
Keywords :

Abstract

This document gives an introduction to the ideas which are being used to set up a simple, factorised, high level model of parts of the ATLAS trigger system. It is intented for a potential newcomer to gain an overview.

It describes the philosophy used to specify "simple" nodes in the system with well defined input and output messages.

This should be read purely as an introduction. Actual details of specification documents and code are all available from the central WWW repository.

Version : 2.0
Date : 24-3-1999
Reference :

1 Purpose of this document and outline of framework

The knub of this document revolves around what one could call a ‘glorified abstract set of APIs’. The rest of the document is just a long way of saying it.

There has recently been a initiative to develop a high level model of the ATLAS data flow, based upon Ptolemy. The names on this document are of those who have indicated they would be willing to contribute to such a Ptolemy based effort.

The purpose of this document is to outline a way of formulating this is a highly specified and documented way intended to make it easy for several people to contribute to this independently, and from geographically separated locations.

The first step of the approach is simply to make sure that every single ‘node’ of the system is specified by a design document which defines in detail its ‘boundary’ and ‘responsibilities’. This consists of:

- Specification of the information content of all of the messages a node may receive and send.
- Specification of the responsibilities of the module upon receipt of each message.

At the highest level of design document these specifications will not in any way constrain the mechanism of transferring messages, and hence they will not imply or require any specific language or message passing mechanism. The reader may now see why this is in a sense a ‘generalised’ or ‘abstract’ API (nothing written here should seem suprising, since one would not conceive of formulating any other distributed software project without first agreeing APIs to define the functionality and interfaces!)

Next, in order to set this in any given language and tool, we then need to agree on a specific message passing mechanism to carry the information specified in the ‘API’s. This is discussed.

Following these steps the code writer may now implement a node to perform these responsibilities in whatever level of internal detail is appropriate. The compatibility with other pieces of software being assured via the first two steps.

One of the most important aspects of this framework is that the documents should be produced as the result of discussion and agreement of all those involved. Such discussion is necessary to fix both the information content of messages and the appropriate level of sophistication required in each node. In general code should only be written following this stage, and conversely the definition stage should not be constrained by any previously existing code. It is hoped that this will lead to a situation where everyone involved ‘owns’ the intellectual side of the work.

For the rest of this document we take the case of a ROB as example. We try to outline the form and type of content of these design documents. Please bear in mind that what is presented below is only intended to show the overall scheme being suggested, therefore the reader should not worry about whether they agree or not with the contents of messages ...etc.. at this stage.

2 Highest level document to specify boundary

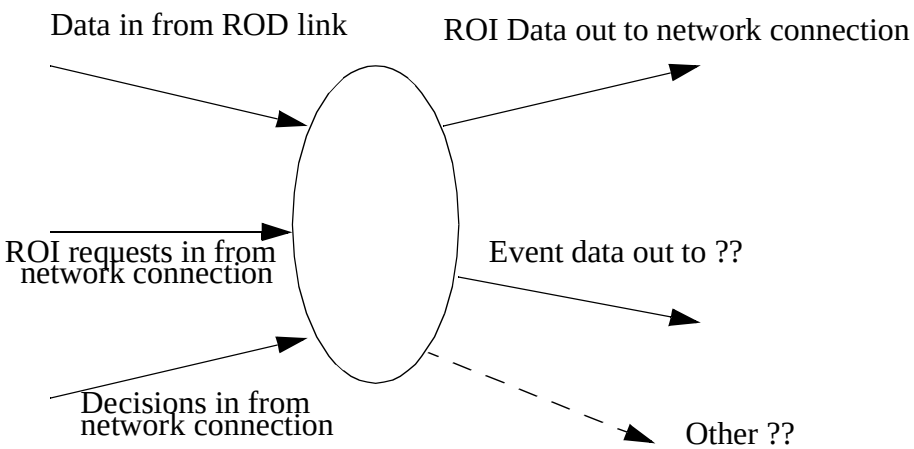
2.1 Preamble

In order to keep the simulation simple we suggest the following:

> Within nodes all actions happen at ‘instants’. In other words when an action is required (i.e. to process some data) then the time taken to do this is calculated, the node waits, and the action is then deemed to have happened at the instant the wait time expires. Although a simplification it is quite likely that this is adequate for a high level model.

2.2 Context

A context diagram is obligatory. In this case hundreds have been drawn before so there is nothng new here:



2.3 Message content

Message name : Data

This message signifies that a block of data has been transferred into the ROB. It is sent by a data supplier (probably the event generation module, but could also be a ROD).

Message name: **`Data`**

Contents {

`EventId`	*simulated event identifier*
`----------`	

`Size`	*number of words transferred in data bloc*
`SourceId`	*something to identify the sender*
`DestId`	*something to identify intended recipient*
.....	*etc...*

}

Responsibilities:

1. Upon reception of this message the ROB should note that it contains this data block.
2. Store this message as it will be requested as an output.
3. If an **`RoiDataRequest`** message for this event is already present in the node then cause the **`RoiData`** message to be sent at a time as as determined by the specific ROB implemetation. Following this the **`RoiDataRequest`** message should be deleted.

2.3.1 Example: "ROI DataRequest"

Message name: **`ROIDataRequest`**

Contents {

`EventId`	*simulated event identifier required*
`ROISpecifier`	*something to specify ROI size (in case of data selection)*
`Size`	*number of words transferred in message*
`SourceId`	*something to identify the sender*
`targetAddress`	*the address to where the data should be sent*
.....	*etc...*

}

Responsibilities:

1. Upon reception of this message the ROB should note that it contains this unsatisfied request.
2. If the **`Data`** message for this event is already present in the node cause the **`RoiData`** message to be sent to the `targetAddress` at an appropriate time. Following this the **`RoiDataRequest`** message should be deleted.

The idea here is that the interface between nodes is unambiguously clear. Someone writing the ROB node must guarantee to accept these messages on these ports, and likewise someone writing a node which sends to a ROB can only use this message set. The aim is to achieve something like the way hardware development works. Once the boundary protocol is defined, the

engineer can implement the insides in any way they wish. The same is true here as the internal structure of the node is now free for whoever writes it to chose, safe in the knowledge that they can bring along their module and it is guaranteed to plug into the simulation.

There is almost certainly more which needs to be added to this level of document. To be discussed...

3 Next level down: documents defining a specific message passing protocol

Up to this point nothing has been written which requires one to know whether you are using C++, Ptolemy

Having decided to use Ptolemy we must now prescribe exactly how a message is to be passed. In general the options are:

A message = an Object method signature (name and Typed argument list)

A message = a pointer to a generic Message object which has all of the services necessary to characterise the message (i.e. Message.Name(), Message.Size()...)

A message = a pointer to one of many object Types, each Type of which represents a specific message. This has the advantage that overloading can be used to avoid if statements.

A message = simple integer encoding of the message into an integer array (euch!!!)

A message = ...

We have chosen to use the third option. This is largely because it is keeps an OO philosophy, and is most compatible with the scheme suggested by Ptolemy.

In a little more detail what we have chosen is to base all messages on the Ptolemy Message class which is designed for this type of use. Our messages indirectly inherit from this, with a new class being defined for each message type.

Thus message passing is achieved by passing different types of object around the system. Each object provides the services required to represent the information which that message should carry in real life.

This is explained in much greater detail in other documents on the main WWW pages, where a reference catalogue of messages is provided.

4 Example

For the best example of a specification document, the reader should look at the “Simple Rob” document available on WWW.