Narval Lexicon

X. Grave
on behalf of the Narval Team

October 12, 2011

Contents

1 A

1.1 Acquisition

An acquisition is the entire process that takes data, manipulates it, makes it available to physicists, Run (13.1) after Run.

1.2 Actors

An actor is a Narval (10.1) process that acts in a consistent way with the state machine (14.4) in a sub system (14.5) domain. There are three kinds of actor which handle part of the Data Flow (4.1) :

- producer

  produces data buffers (2.1)

- intermediary

  manipulates data buffers as input and output

- consumer

  consumes data buffers

A special actor is available : the standalone actor. It doesn't handle data, but can execute tasks synchronously with the others Narval actors.

1.3 Ada

Ada is an ISO standardised programming language. It provides tasking and mutual exclusion on data and since 1995 it also provides object oriented programming and distributed computing facilities. Since 2005, interfaces “à la” java are available in the language.

1.4 Annex E of Ada 95

As Ada 95 is an ISO standard and in order to avoid too much pressure on compiler manufacturers, the language proposes optional annexes. The DSA[1] allows one to develop a code that will run on heterogeneous hardware and operating systems across a network without any explicit network references. The GNAT compiler proposes the DSA as a set of tools and libraries : Poly-ORB.

[1]Distributed System Annex

1.5 API

API stands for Application Programming Interface.

1.6 aws_shell

This process is the bridge between the Narval (10.1) system and the rest of the world. It provides a Web Services interface, which uses the SOAP (14.3) protocol implemented by the AWS (1.7) library.

1.7 AWS

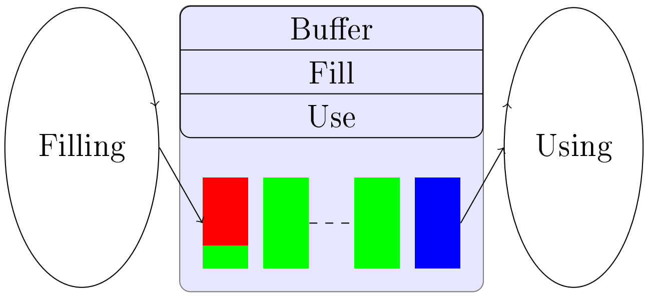
AWS[2] stands for Ada Web Services. AWS is a set of libraries that provides embedded web server and web services.

2 B

2.1 Buffer

A buffer in the Narval framework is a set of memory blocks that is protected from concurrent access. Indeed many tasks access data in Narval (10.1) :

- Network input and output tasks
- Each actor (1.2) working task (consumption, manipulation and production)



[2] AWS @ Ada Core

3 C

3.1 central_log

The central log process collects trough Ada 95 annex E (1.4) all the logs produced by the Narval system (10.1). It relies on the log4Ada (9.1) library.

3.2 clean_narval

This script is used in Narval (10.1) environment to clean a Narval System in error state (14.4).

3.3 consumer

A consumer is an actor that uses data from the Narval Data Flow (4.1). There are many different kinds of utilisation :

- Histograms

  ROOT, GRU (7.6), Cvisu (3.4)

- Data storage

  IN2P3 format, RAW

- Calibration

- etc...

3.4 Cvisu

This program is used to display and do computation on histograms. It was an OASIS (11.1) companion. It uses System V RPC to communicate (messages queues, shared memory, ...).

4 D

4.1 Data Flow

The Data Flow is the collection of paths the data are following along processes in Narval, similar to the rain streaming along a mountain and reaching a delta. The multiple paths follow all the same general direction and no turning back is possible. The Data Flow is determined by the topology (15.1).

4.2 Debian

Debian aims to be an universal operating system. It is an Open Source Project. It mainly uses the Linux kernel. Narval (10.1) is packaged for it. Since Ubuntu (16.1) is built upon Debian, it also includes Narval packaging.

5 E

5.1 Event Builder

The event builder functionality is performed by a specialised intermediary (1.2). It is developed in Ada-only since it needs access to the full buffer (2.1) and tasking API present in actors (1.2).

6 F

6.1 filter

The filter is a one-on-one Narval intermediary. It consumes a buffer and produces one. A generic version is available which can load a C/C++ shared library (14.2).

6.2 force_clean_narval

This script is similar to the clean_narval (3.2) one, except that it works also when the sub system isn't in error state.

7 G

7.1 generic_ada_actor

This actor (1.2) loads an Ada shared library (14.2). This library has to implement an Ada interface : Actor_Interface. This interface can be found in the plugin_interfaces.ads file from narval.actors.generic.ada branch. Unlike the C and C++ shared libraries used by the others generic actors (7.2, 7.3, 7.4) the library can implement any type of actors (producer, intermediary, consumer) and can have more than one buffer (2.1) as input and more than one buffer as output.

7.2 generic_consumer

The generic_consumer actor (1.2) loads a C or C++ shared library (14.2) that complies with this library API. It consumes one buffer input only. A typical use is data storage.

7.3 generic_filter

The generic_filter actor (1.2) loads a C or C++ shared library (14.2) that complies with this library API. It consumes one buffer input only, processes it and then produces one output buffer. A typical use is data filtering : shaping, and so on...

7.4 generic_producer

The generic_producer actor (1.2) loads a C or C++ shared library (14.2) that complies with this library API. It produces one buffer output only. A typical use is electronic data collecting or file reading.

7.5 generate_narval_conf

This script is used generally only once in order to produce a file named narval.conf that will be used by the Annex E (1.4) environment variable POLYORB_CONF. It contains information in order to allow the PolyORB framework to organise itself on the network.

7.6 GRU

GRU stands for GANIL ROOT Utilities. It is a framework to facilitate the analysis of data both online and offline. In Narval (10.1), it is generally used to separate histograms filling (in the Data Flow, 4.1) from histograms display.

8 I

8.1 intermediary

The intermediary actors (1.2) have inputs and outputs buffers (2.1) and uses the input ones to calculate the output ones. They can be seen as $N \times M$ routing matrix that can manipulate the data they transmit.

9 L

9.1 Log4Ada

This library (included in the Debian distribution (4.2)), is a logging facility that aims to be compatible with the log4j framework.

10 N

10.1 Narval

Narval is a framework to develop Data Acquisition Systems. It eases the distribution across a network for the user of simple to complex systems. One can develop simple plugins as shared libraries (14.2) in C, C++ and Ada for the Narval generic actors (7.1, 7.4, 7.3, 7.2). It is also possible to inherit (in an Oriented Object way) from the actor object and fully customise an Ada actor for special needs (typically an Event Builder : 5.1).

10.2 narval_launch

A script that starts all the principal processes of Narval (10.1) :

- narval_naming_services (10.3)
- aws_shell (1.6)
- central_log (3.1)

10.3 narval_naming_services

The narval_naming_services process is used to store a list of available narval sub systems (14.5). When a sub_system_coordinator (14.6) starts, it registers in this process, and when it finishes, it unregisters.

10.4 narval_shell

A shell command utility to send orders to Narval sub systems trough aws_shell (1.6).

11 O

11.1 Oasis

It is the ancestor of Narval. It was a mono bloc client/server system where the embedded software was running with VxWorks and the storage and online analyse was running on a Solaris Box. Cvisu (3.4) was used to display histograms.

12 P

12.1 producer

A producer is an actor that gets data from different kinds of devices :

- Electronic boards

  VME, PCI, Ethernet, ...
- Files

  IN2P3 format, RAW
- etc...

Then it injects the data into the Narval Data Flow (4.1).

13 R

13.1 Run

A Run is a set of parameters (for electronics and algorithms), experimental data delimited by the two following events : start and stop. Usually, Runs are indexed by experiment name and a number.

14 S

14.1 setup_narval_keys

A shell command to setup ssh keys to allow a remote connection without a password. For more information : "man setup_narval_keys".

14.2 Shared library

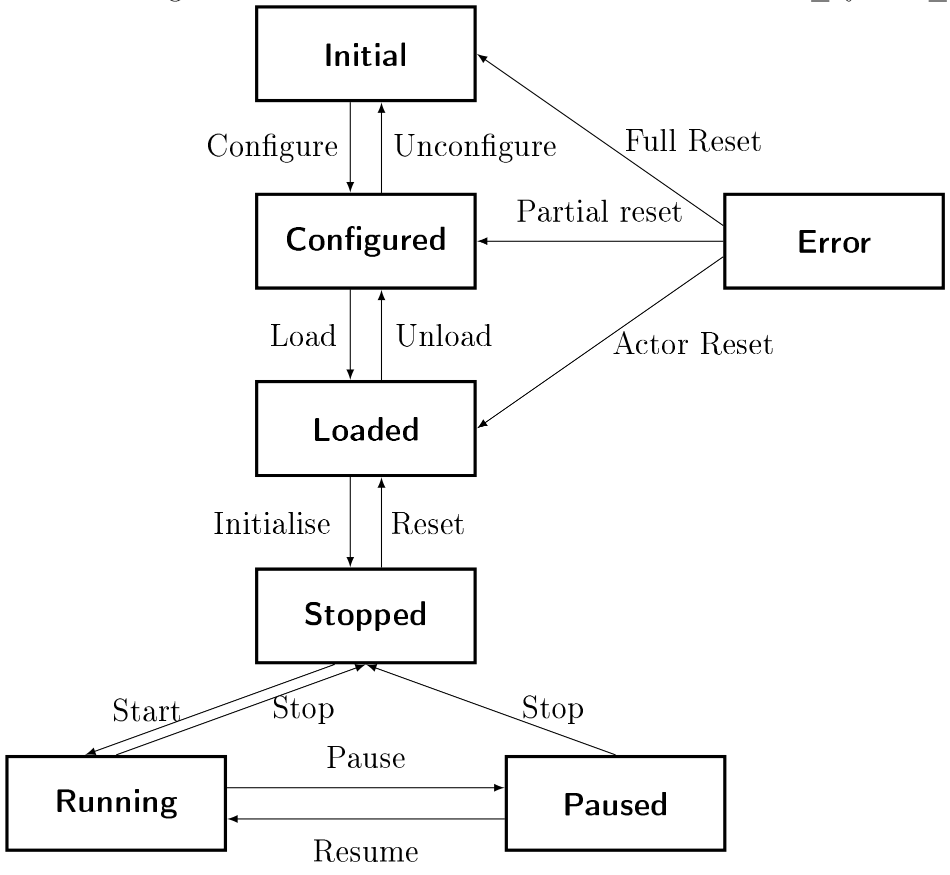
A shared library is a set of binary code objects files that can be dynamically loaded by processes.

14.3 SOAP

SOAP[3] stands for Simple Object Access Protocol. It is a RPC protocol based on XML.

14.4 State machine

Here is a diagram of the state machine embedded in each sub_system_coordinator (14.6).



[3]SOAP @ WikiPedia

14.5 sub_system

A Narval sub system is a set of processes : a sub system coordinator (14.6) and actors (1.2) that runs in coordination (thanks to the state machine 14.4) to acquire, process, store data from experiments.

14.6 sub_system_coordinator

The sub system coordinator is the process that contains the state machine (14.4) and also the list of actors (1.2) that will be used to do the job. It distributes all the orders synchronously to the actors.

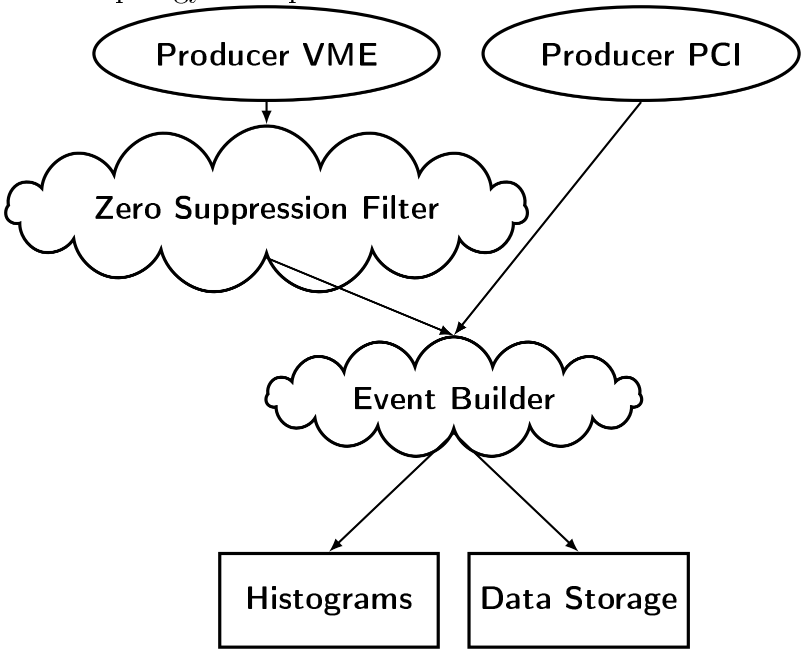
14.7 system

The system, is the Narval (10.1) framework in action.

15 T

15.1 Topology

A Narval (10.1) topology is the set of actors that handle the Data Flow (4.1). Here is a topology example :



16 U

16.1 Ubuntu

Ubuntu is an ancient Zulu word that mean : “I don’t know how to install Debian”.