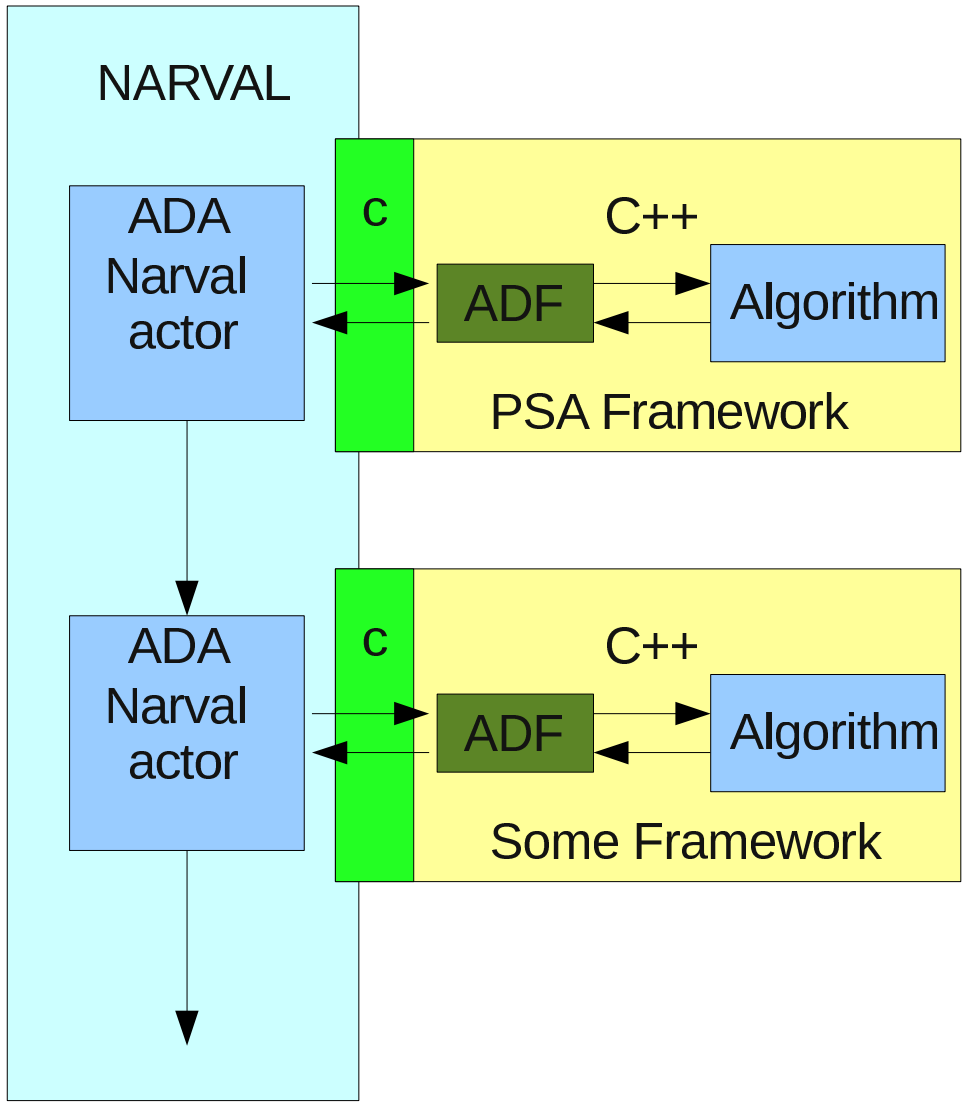
NARVAL implementation of a PSA framework and a NARVAL emulation for code development

Joa Ljungvall

July 10, 2008

Why a PSA framework?

- Remove complications of interfacing with Narval/ADF from the PSA developer to me;)



What does it look like?

- Using c++ inheritance - base class PSAFilter
- ADF provides the connection to Narval

Base class provides the services of talking to ADF/Narval and some virtual members to implement for each algorithm to do the work.

1. `virtual Int_t InitDataContainer()` - For loading databases of pulseshapes, base class provides pointer for this
2. `virtual Int_t ResetDataContainer()` - Maybe we do a processreset
3. `virtual Int_t Process()` - The actual PSA code goes here

How is it used?

Do something like:

```
struct GridSearchData {
baseSim *base;
float *metrica;
};

class PSAFilterSimpleGrid :  public PSAFilter
{...
Int_t Process();
//In here your open your data file etc;
Int_t InitDataContainer();
//Here you close data file etc.
Int_t ResetDataContainer();
...};
```

How is it used?

- In `InitDataContainer()` do

```
...
fDataContainer = new GridSearchData;
.
.
.
((GridSearchData*)fDataContainer)->base = base;
```

How is it used?

- For `Process()` there are a few things to remember:
  1. When called the traces can be found in "classical" short arrays, `*CoreTrace_pp` and `*SegmentTraces_pp[36]`.
  2. It should end with a piece of code like:

```
 ahit = new ADF::PSAHit;
ahit->SetE(PtoExp->NetCharge[iii][1]);
ahit->SetX(PtoExp->pto.x);
ahit->SetY(PtoExp->pto.y);
ahit->SetZ(PtoExp->pto.z);
AddHit(ahit);
```

     This to put the hit into the ADF data stream.

If you try this you will find it simple;)

A Narval emulator

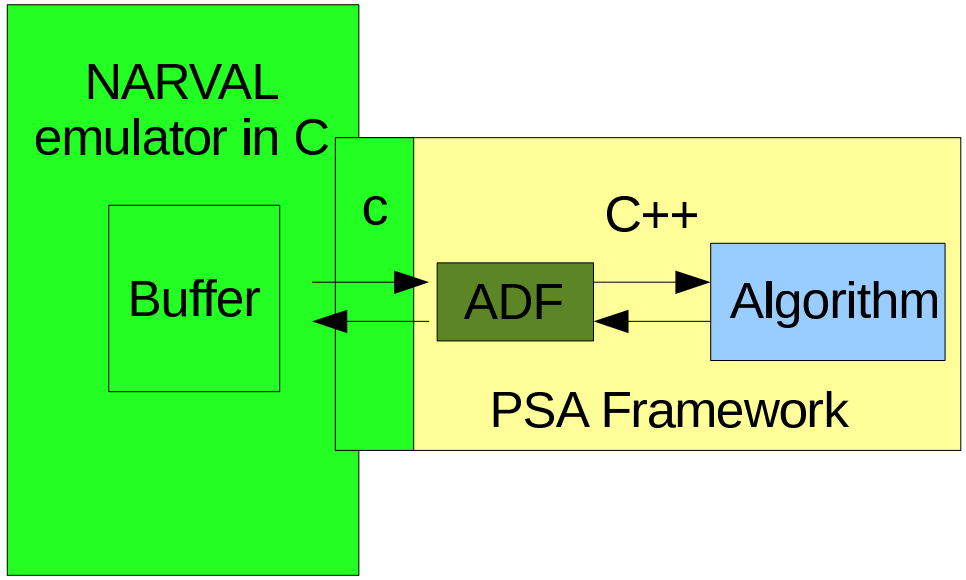
Why?

- Develop code
- Debug your code
- Profile your code

What does it offer?

- A “linear” Narval topology
- 1 producer, X filters, 1 consumser
- Can be debugged unsing standard c/c++ tools on any platform

- how?



- how?

From a more technical point of view:

```
{
void (*process_block_filter[MACTORS-2])(...);
void *actors [MACTORS];
.
.
process_block_filter[ii-1] = dlsym(actors[ii],
"process_block");
.
}
```