May5th, 2010
Y. Chao

Some Thoughts on High Level Application Demands

Many already addressed in the past two days $\Rightarrow$ Not repeated here

Only questions, No answers

Why XAL?

- Infrastructure built to modern software engineering standard
- Structure & organization – well thought out
- Suite of tools – Versatile, capable, ready to apply
- Extensibility for developers

But this is a view from someone like me (a Sales Rep.?)

Who are the real customers?

- Local code developer (Software, Physicists, ……)
- End user (Operators, Physicists, System experts, ……)

The speed at which the customer is lost can be very fast

- Too many hoops to jump through in developing the application
- Too many hoops in running the application
- Not intuitive enough
- Too time consuming to execute
- Can't deliver what's advertised / what's really needed
- Does not seem indispensable (didn't make my life any different)
- Process crash / conflicts, erratic behaviour
- Poor documentation / online help
- Poor support
- Wrong outcome!

Negative "word of mouth" is all it takes to quickly kill an actually very good tool. Second chance rarely happens.

So what will make a tool (XAL) gain traction?

The strength of XAL must not be overshadowed by superficial “nuisances” (if at all)

To be determined: Core feature, or site-specific extension?

Many high level issues, but enabling provisions may need be made at low level

For Code Developers:

- Software engineers
  - Tool is structured up to software engineering standards
  - Relatively streamlined development protocols
  - Maintainability – How can this be built-in for high level apps?
    - Avoid interdependency – Always possible? Always desirable? How to ensure structure integrity?
    - Synchronized upgrade of entire hierarchy – Low to high level, model, file structure, database, ……
    - How about Jython & Matlab scripts?
- Physicists
  - Relatively streamlined development protocols
  - Scalability
  - Efficient algorithm-to-prototype turn-around
    - Competent math toolbox
    - Competent and “comprehensive” modeling capability
    - Competent and “comprehensive” logistic functions (plotting, archiving,…)
    - Ability to efficiently implement new devices and processes in the model
    - Algorithm testing platform (realistic machine simulation, realistic diagnostic/control simulation, error representation, ……)
  - Efficient machine experiment execution through the tool
    - Massive data collection / archiving
    - Flexible implementation of multiple control point changes in multiple steps (in user defined pattern)
    - Full event reconstruction offline – further facilitates algorithm testing
  - More demand on the model
    - Main source of machine model information – most logical place to obtain physics related to real machine – maybe phased

For End Users:

Two modes of end users (by task, not job title):

Not always the same objectives and preferences.

- Operator mode
  - Deliver beam. Well defined path, minimal distraction
  - Absolutely free of bugs or likelihood to crash
  - Easy to use
  - The faster, the better
- Physicist mode
  - Understand machine / Commission new methods. Undefined path, maximal information, tweak/grope on the fly
  - Mainly demand on the high level apps design, but low level robustness is critical
  - Modularity – Ability to swap in/out utility/algorithm, input/output modules efficiently
  - Some of the above for developers applies here.

My (biased) Message

- Keep high level apps in mind while developing low level infrastructure.
- Documentation and support will go a long way.
- Most extreme and possibly competing demands come from physicists as developers and operators as end users.
- A “killer package”, overcoming the inertia in both groups at the same time, <u>may be what we have to do.</u>
-