SPS2001

Towards a common control system?

- Motivation is fairly clear
  - Economics
  - And it gives us a chance to work together and be nice to each other

- Common control system is a nice idea but we have:
  - History
  - Personality
  - Management

Pragmatism might be useful…

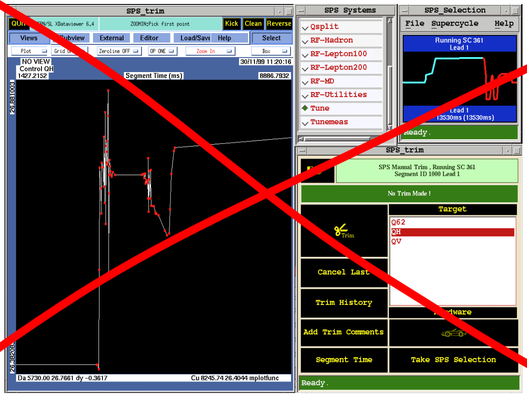
SPS re-write

- Fast supercycle changes for the LHC era
- Interlocks
  - cycle dependent interlocks
- Improved trim functionality
- Rationalisation of historically crowded control system
  - Heterogeneous
  - Out-moded technology e.g. C-tree

Reasonable justification for a re-write

Situation

- We know how to operate the SPS
- We have a working, and quite successful, control system



SPS Manual Trim , Running SC 361 Segment ID 1000 Lead 1

No Trim Made !

Ready.

Ambitious attempt to re-implement from top-to-bottom…

This poses some challenges…

So If We're Going to Do It Anyway

- Let's at least try and come up with an approach and solutions that are acceptable and re-usable

- And in this sense - as we look down the transfer lines towards the LHC - use SPS2001 as a test bed for these common solutions

- Many of the control challenges are similar after all

Challenges I

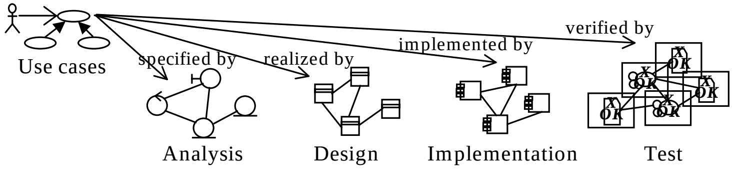
- OO: kicking and screaming into the 21st century
  - Analysis and design:
    - Unified Software Development Process (USDP)
    - UML Ł
    - Case tool – Rational Rose
  - The languages:
    - Java, XML, SQLJ, JDBC
  - The IDE: Jdeveloper, Jbuilder…, GUIs
  - Other tools: Jpublisher, Swing etc. etc
  - Communication
    - Inter Java RMI etc
  - Persistance
    - How to do OO to RDBMS mapping?
    - Which tools to use, if any? Ł

All this does have its advantages, of course

And we really should be doing what everybody else is doing

Ł Analysis and Design

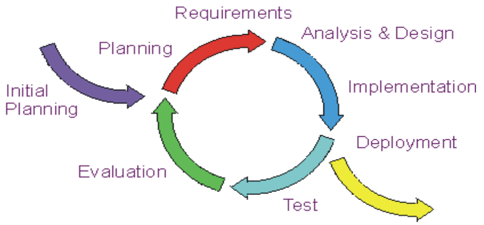
- Establish requirements
- Learn how to establish requirements
- Analysis
- Learn how to analyse the problem & use the language
- Design
- Explore architectural choices and address issues arising
- Explore technical choices



Widely accepted method…

Ł Analysis & Design

The method is clearly re-usable and if used rigorously provides a sound, well-understood, well documented understanding of the problem and a well designed system.



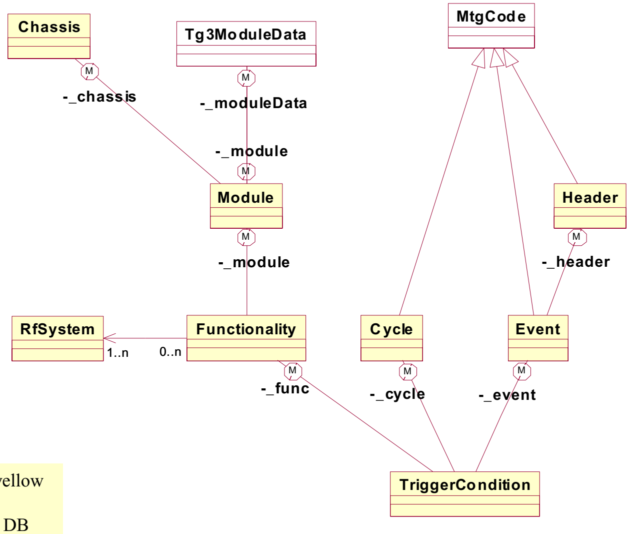
If we get it right, a lot of the concepts, if not the classes, will be re-usable e.g. in the domain of parameter maintenance

“Models if not done properly: vague, sloppy and inaccurate”

70% of projects: Level 1: Chaotic, only saved by hero

25% of projects: Level 2: repeatable but no insight

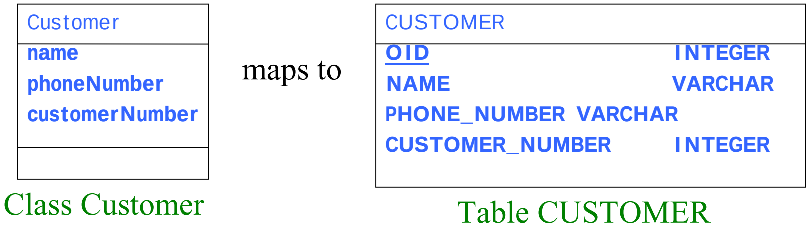
Ł Persistence



Classes in yellow need to be persisted in DB

Ł Mapping objects to RDB tables

- Attributes **map to database** columns
  - **each attribute maps to zero or more columns**
- Classes **map to** tables
  - **each class maps to one or more tables (can use table joins to handle inheritance)**
- **Every object has a unique** object ID
  - **the** OID **uniquely identifies the object (at least within its own class)**
  - **OIDs are used as** primary keys **in the database**



Class Customer

Table CUSTOMER

Ł Mapping OO to RDBMS

- **Poses a number of problems being wrestled down out in the world**
- **Oracle seems a natural choice for the SPS to replace C-tree as a settings and configuration repository.**
- **Experience gained in this domain clearly of future use**
- **Solutions found in this domain clearly of potential future use**

Challenges II - middleware

- **A common solution to provide command/response and publish and subscribe would be nice**
- **Um.. Synchronous and asynchronous**
- **Oh and I need to configure my instrumentation at the same time as I request a measurement**
- **… but what if someone else is subscribing to that measurement..**
- **User options, filters, cycle dependency etc…**
- **Don't swamp my front-end with your device server**
- **Commercial solution (maybe, maybe not)**
- **Topics on subscribe**
- **And could a device tell me what it's capable of…**
- **Oh and contracts**

See Vito's talk…

Challenges II – Front-ends

- **Common Front-End:**
  - **hardware architecture**
  - **software architecture**

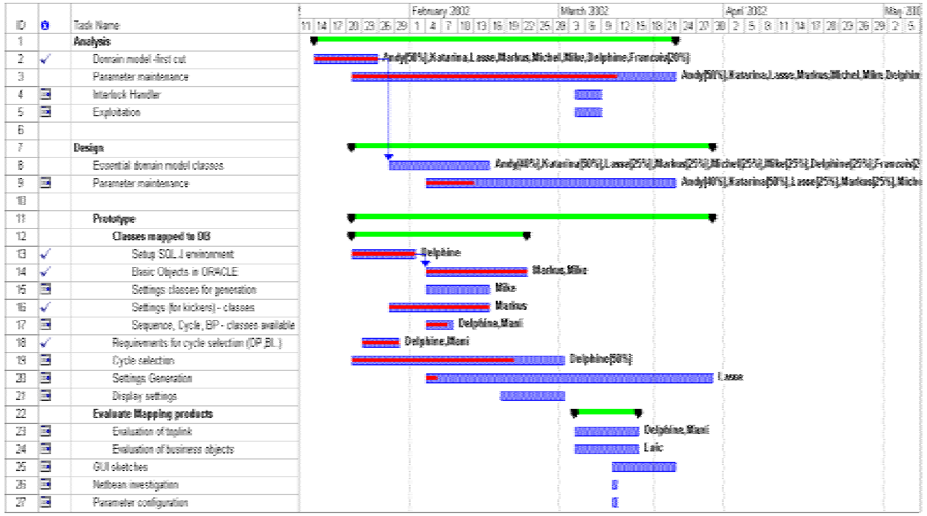
are desirable… see Pierre's talk

Challenges III

Project Bloody Management

'nuff said…

Progress

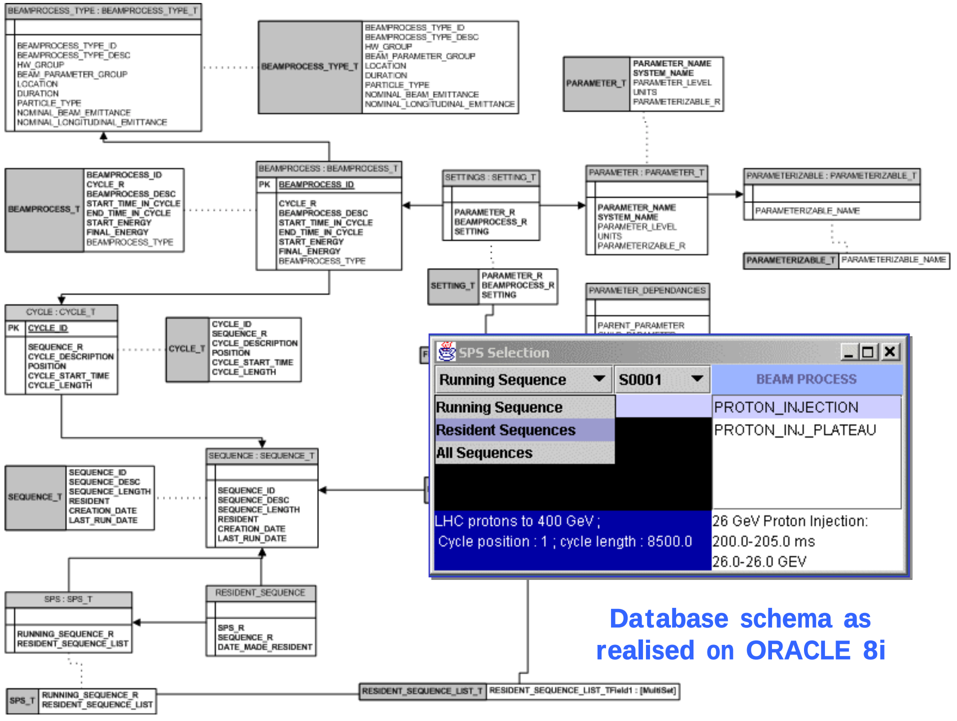


ID		Task Name
1		**Analysis**
2	✓	Domain model -first cut
3		Parameter maintenance
4	▤	Interlock Handler
5	▤	Exploitation
6		
7		**Design**
8		Essential domain model classes
9	▤	Parameter maintenance
10		
11		**Prototype**
12		**Classes mapped to DB**
13	✓	Setup SQLJ environment
14	✓	Basic Objects in ORACLE
15	▤	Settings classes for generation
16	✓	Settings (for kickers) - classes
17	▤	Sequence, Cycle, BP - classes available
18	✓	Requirements for cycle selection (DP,BL)
19	▤	Cycle selection
20	▤	Settings Generation
21	▤	Display settings
22		**Evaluate Mapping products**
23	▤	Evaluation of toplink
24	▤	Evaluation of business objects
25	▤	GUI sketches
26	▤	Netbean investigation
27	▤	Parameter configuration

February 2002
March 2002
April 2002
May 200
11 14 17 20 23 26 29 1 4 7 10 13 16 19 22 25 28 3 6 9 12 15 18 21 24 27 30 2 5 8 11 14 17 20 23 26 29 2 5
Andy[50%],Katarina,Lasse,Markus,Michel,Mike,Delphine,Francois[30%]
Andy[50%],Katarina,Lasse,Markus,Michel,Mike,Delphin
Andy[30%],Katarina[50%],Lasse[25%],Markus[25%],Michel[25%],Mike[25%],Delphine[25%],Francois[
Andy[40%],Katarina[50%],Lasse[25%],Markus[25%],Miche
Delphine
Markus,Mike
Mike
Markus
Delphine,Mani
Delphine,Mani
Delphine[50%]
Lasse
Delphine,Mani
Loic

Analysis and design

- It has been a steep learning curve…
- Consultancy has been necessary and has helped (and is helping…)
- Got moving on parameter maintenance domain
  - Pushed through analysis into design
    - More-or-less acceptable domain model
    - First cut PMD analysis finished
    - Moving into design stage
- Prototyping and gaining experience in mapping our classes on to an RDBMS:
  - Prototyped sequence, cycle etc as persistent objects using SQLJ, object related features of Oracle 8i
  - Given this: cycle selection prototype (Delphine)
  - Store and retrieve of kicker settings (Markus)
  - Settings generation to persistent objects. (Lasse)



BEAMPROCESS_TYPE : BEAMPROCESS_TYPE_T
BEAMPROCESS_TYPE_ID
BEAMPROCESS_TYPE_DESC
HW_GROUP
BEAM_PARAMETER_GROUP
LOCATION
DURATION
PARTICLE_TYPE
NOMINAL_BEAM_EMITTANCE
NOMINAL_LONGITUDINAL_EMITTANCE

BEAMPROCESS_TYPE_T
BEAMPROCESS_TYPE_ID
BEAMPROCESS_TYPE_DESC
HW_GROUP
BEAM_PARAMETER_GROUP
LOCATION
DURATION
PARTICLE_TYPE
NOMINAL_BEAM_EMITTANCE
NOMINAL_LONGITUDINAL_EMITTANCE

PARAMETER_T
PARAMETER_NAME
SYSTEM_NAME
PARAMETER_LEVEL
UNITS
PARAMETERIZABLE_R

BEAMPROCESS_T
BEAMPROCESS_ID
CYCLE_R
BEAMPROCESS_DESC
START_TIME_IN_CYCLE
END_TIME_IN_CYCLE
START_ENERGY
FINAL_ENERGY
BEAMPROCESS_TYPE

BEAMPROCESS : BEAMPROCESS_T
PK BEAMPROCESS_ID
CYCLE_R
BEAMPROCESS_DESC
START_TIME_IN_CYCLE
END_TIME_IN_CYCLE
START_ENERGY
FINAL_ENERGY
BEAMPROCESS_TYPE

SETTINGS : SETTING_T
PARAMETER_R
BEAMPROCESS_R
SETTING

PARAMETER : PARAMETER_T
PARAMETER_NAME
SYSTEM_NAME
PARAMETER_LEVEL
UNITS
PARAMETERIZABLE_R

PARAMETERIZABLE : PARAMETERIZABLE_T
PARAMETERIZABLE_NAME

PARAMETERIZABLE_T
PARAMETERIZABLE_NAME

SETTING_T
PARAMETER_R
BEAMPROCESS_R
SETTING

PARAMETER_DEPENDANCIES
PARENT_PARAMETER

CYCLE : CYCLE_T
PK CYCLE_ID
SEQUENCE_R
CYCLE_DESCRIPTION
POSITION
CYCLE_START_TIME
CYCLE_LENGTH

CYCLE_T
CYCLE_ID
SEQUENCE_R
CYCLE_DESCRIPTION
POSITION
CYCLE_START_TIME
CYCLE_LENGTH

SEQUENCE : SEQUENCE_T
SEQUENCE_ID
SEQUENCE_DESC
SEQUENCE_LENGTH
RESIDENT
CREATION_DATE
LAST_RUN_DATE

SEQUENCE_T
SEQUENCE_ID
SEQUENCE_DESC
SEQUENCE_LENGTH
RESIDENT
CREATION_DATE
LAST_RUN_DATE

SPS : SPS_T
RUNNING_SEQUENCE_R
RESIDENT_SEQUENCE_LIST

RESIDENT_SEQUENCE
SPS_R
SEQUENCE_R
DATE_MADE_RESIDENT

SPS_T
RUNNING_SEQUENCE_R
RESIDENT_SEQUENCE_LIST

RESIDENT_SEQUENCE_LIST_T
RESIDENT_SEQUENCE_LIST_TField1 : [MultiSet]

PROTON_INJECTION
PROTON_INJ_PLATEAU

LHC protons to 400 GeV ;
Cycle position : 1 ; cycle length : 8500.0

26 GeV Proton Injection:
200.0-205.0 ms
26.0-26.0 GEV

Database schema as realised on ORACLE 8i

Mapping

Attempting to wrestle this one to the ground at the moment…

- Castor JDOs evaluated
- Toplink evaluated
- Business Components for Java evaluated
- Played with JPublisher
- Database Objects created by hand, database gateway classes created and cycle selection prototyped
- Database Objects created for kicker settings archival and retrieval

Seeking an appropriate technical solution for our requirements

Tools, Services, Components…

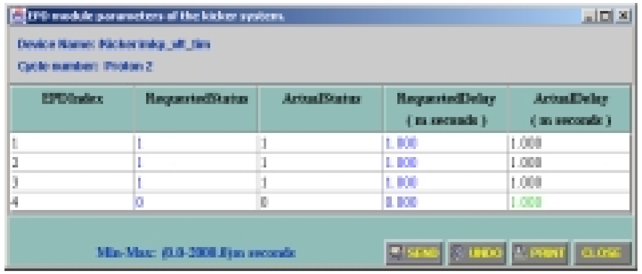
In place and in use…

- Code style: JIndent
- Graphical User Interfaces: StopMi
- IDE: Jbuilder and JDeveloper
- Version control: Razor
- Case tool: Rational Rose
- Report generation from Rose: SoDA
- Error Handling: log4J evaluated
- XML: widely used
- Testing: tools evaluated
- Components: e.g. Java Dataviewer

Message: OO firmly embraced…

XML

```
<table id="EPD" format="long" depth="4" >
  <comment text="EPD module parameters of the kicker system." />
  <permission expert="RWC" control="R" operator="R" public="R" />
  <column name="EPDIndex" accessibility="None" minimum="1" maximum="4" />
  <column name="RequestedStatus" accessibility="RW" minimum="0" maximum="1" />
  <column name="ActualStatus" accessibility="R" minimum="0" maximum="1" />
  <column name="RequestedDelay" accessibility="RW" unit="seconds" scale="-6“
          precision="1000" minimum="0" maximum="2000000" />
  <column name="ActualDelay" accessibility="R" unit="seconds" scale="-6“
          precision="1000" minimum="0" maximum="2000000" />
</table>
```



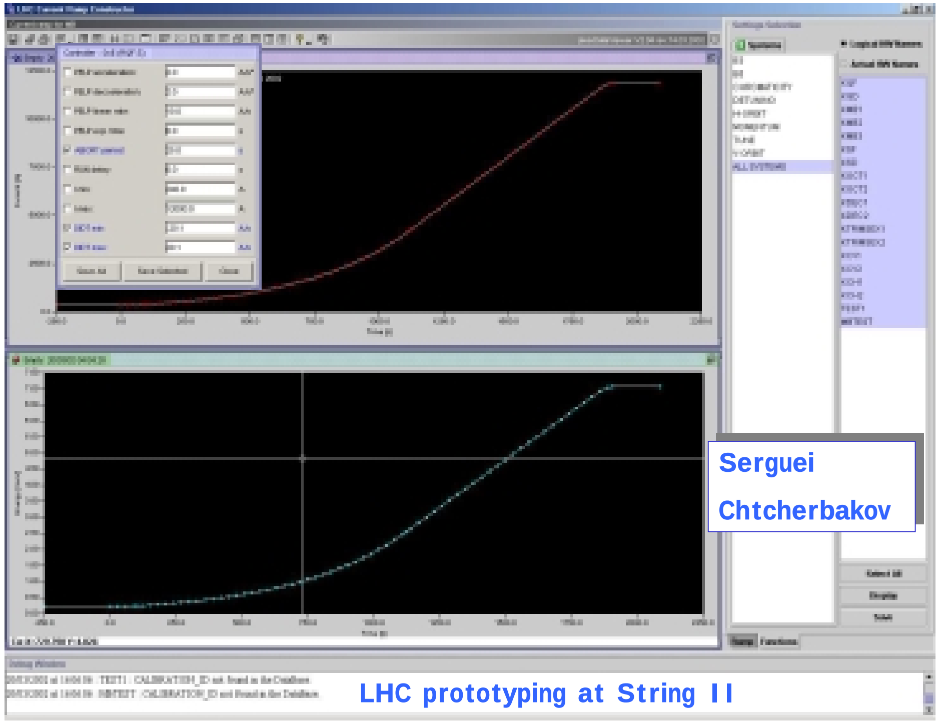
EPD module parameters of the kicker system.

Device Name: Kickerinkp_alt_tim

Cycle number: Proton 2

EPDIndex	RequestedStatus	ActualStatus	RequestedDelay (m seconds)	ActualDelay (m seconds)
1	1	1	1.000	1.000
2	1	1	1.000	1.000
3	1	1	1.000	1.000
4	0	0	1.000	1.000

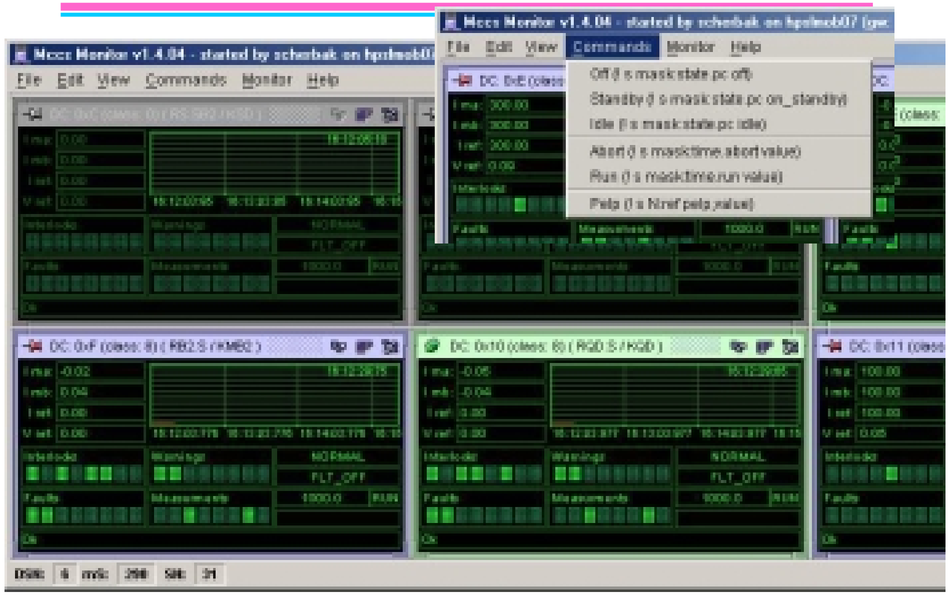
Min-Max: [0.0-2000.0] m seconds



Serguei Chtcherbakov

LHC prototyping at String II

LHC Prototyping – String II

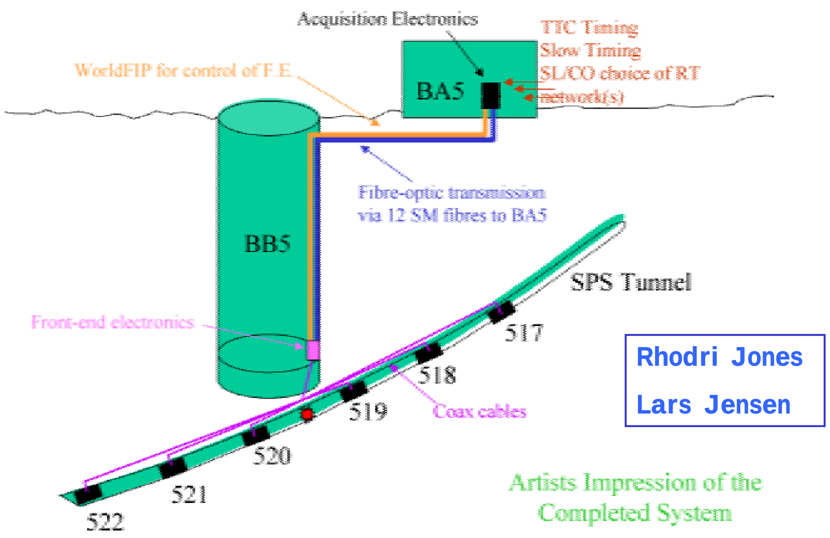

Mccs Monitor v1.4.04 - started by scherbak on hpslmob07
File Edit View Commands Monitor Help
Off () s mask state pc off)
Standby () s mask state pc on_standby)
Idle (/ s mask state pc idle)
Abort () s masktime abort value)
Run () s masktime run value)
Pelp () s Nref pelp value)
DC: 0xF (class: 8) (RB2.S / KMB2)
DC: 0x10 (class: 8) (RQD.S / KQD)
DC: 0x11 (class
I max
I min
I ref
V ref
Interlocks
Warnings
NORMAL
FLT_OFF
Faults
Measurements
1000.0
RUN
Ok

Local Orbit Feedback Prototype

Proposal:

- Install 6 pickups around point 5 of the SPS and equip them with LHC front-end electronics (WBTN)
- LHC data acquisition system (DAB - version 2 prototype from Triumph)
- Link to PCR & pull local orbit measurements to high level and display position in real-time (10 Hz)
- Closed loop at high level with RT connection to power converters to prototype local orbit feedback

Local Orbit Feedback Prototype



Rhodri Jones

Lars Jensen

Artists Impression of the Completed System

Conclusions

- Clear that as we move into the LHC era we don’t have the resources to support multiple solutions to similar technical problems. Need to settle on:
- Analysis and design approach:
  - USDP, UML, Rational Rose etc.
- Development environment:
  - JAVA, IDE, ORACLE, GUIs
  - Persistence layer
  - Version control etc.
  - Supporting applications: dataviewer, fixed displays…
  - Use of EJBs, application servers or whatever...
- Operational environment
  - Operating system (LINUX?)

Conclusions

- Try and move towards a common Middleware
- Front-end Framework
- Explore novel requirements of LHC such as real-time
- Operational tools
  - Sequencer, Logbook, Console manager…
- Services
  - Alarms, Logging, Post mortem, Interlocks…
- Plus, with a bit of luck, some application software…

We're trying to come up with acceptable solutions appropriate for the transfer lines and the LHC…

and before promising the earth we have to prove that it works… watch this space.

Acknowledgements

- Markus Albert
- Andy Butterworth
- Francois Chevrier
- Serguei Chtcherbakov
- Guy Crockford
- Loic D'Heveder
- Delphine Jacquet
- Mani Jorda Garcia
- Michel Jonker
- Lasse Normann
- Katarina Sigerud
- Jorg Wenninger