Software Process in the ATLAS Back-end DAQ

Autumn Trigger DAQ Workshop
Marseille, October 1997

General Approach

- project phases
  - collect requirements (ESA-PSS05 style URD produced)
  - identify common issues (e.g. data storage, graphics, communication etc.)
  - perform pre-design investigations into candidate technologies/techniques
  - develop high-level design
  - detailed-design, implementation and unit testing
  - integration
  - deployment
- principles
  - rapidly evolving software market (e.g. Java, UML, UNIX/WNT)
  - adhere to relevant standards (e.g. OMG, ODMG)
  - use commercial software wherever possible
  - rely on other projects for specific areas (e.g. RD45 persistence)
  - concentrate development effort on ATLAS-DAQ specific items
  - use common solutions across all components of the backend
  - requirements, analysis & design most important aspects to get to 2005

BackEnd components

Run control	controls DAQ configuration and data taking operations
Configuration databases	define all aspects of the DAQ configuration
Message reporting system	report/capture of information messages
Information service	general purpose information exchange
Process manager	basic job control of programs
Status display	shows current status of data taking to the shift operator
Partition/resource manager	allows concurrent data taking activity
Test manager	bank of functionality tests for DAQ components
Diagnostics package	uses tests held in the test manager to diagnose problems
Run bookkeeper	electronic tape log book
Monitoring & event display[a]	access to sampled data for analysis and quality checking
Data and event viewing[a]	facility for viewing event data and sets of histograms

core

data-flow

detector

[a] online aspects only (in conjunction with data-flow group)

BackEnd DAQ status: October'97

component	*require-ments*	*design*	*imple-ment.*	*integra-tion*	*institutes*
Run control	██████████	██████████	███████░░░	░░░░░░░░░░	CERN, IN2P3-Marseille, Sheffield
Configuration databases	██████████	██████████	████████░░	░░░░░░░░░░	CERN, PNPI
Message reporting system	██████████	██████████	████████░░	░░░░░░░░░░	CERN, IAP-Bucharest, PNPI
Information service	██████████	██████████	█████████░	░░░░░░░░░░	CERN, IAP-Bucharest, PNPI
Process manager	██████████	██████████	█████████░	░░░░░░░░░░	IN2P3-Marseille
Status display	██████████	█████░░░░░	░░░░░░░░░░	░░░░░░░░░░	IAP-Bucharest, IN2P3-Marseille
Partition/resource manager	████████░░	████████░░	░░░░░░░░░░	░░░░░░░░░░	JINR-Dubna
Test manager	██████████	███░░░░░░░	░░░░░░░░░░	░░░░░░░░░░	NIKHEF
Diagnostics package	█████░░░░░	░░░░░░░░░░	░░░░░░░░░░	░░░░░░░░░░	
Run bookkeeper	██████████	████████░░	██░░░░░░░░	░░░░░░░░░░	LIP
Monitoring & event display[a]	█████░░░░░	░░░░░░░░░░	░░░░░░░░░░	░░░░░░░░░░	
Data and event viewing[a]	█████░░░░░	░░░░░░░░░░	░░░░░░░░░░	░░░░░░░░░░	

[a] online aspects only (in conjunction with data-flow group)

ATLAS Back-end DAQ

adopted technologies

• StP/OMT & Booch	OO method and CASE tool
• FrameMaker/WebMaker	documentation system
• Objectivity ODBMS	ODBMS for long-term storage
• Tools.h++	general C++ utilities and simple persistence
• Corba/ILU	inter-process communication
• ACE	portable C++ interface to operating system
• Java/Motif	graphics for status display and editors
• X-Designer	cross-platform GUI development and testing
• CHSM	finite state machines in C++
• SRT	configuration management

Back-end DAQ: Definition of requirements

- Deliverables

    To produce a user requirements document

    Define a work-plan for the next phase

- Organisation

    Organised as a working group (19: DAQ + detector reps.)

    Used ESA-PSS05 Framemaker template from ECP/IPT group

- Duration

    4 months (Jan-Apr'96)

- Review

    URD announced at Trigger/DAQ meeting of March 1996 ATLAS week

    Comments (very few) received and incorporated in the URD

    We also produced a summary document: no specific requirements but shorter and easier to read

    Visited LEP experiment sites to discuss back-end issues and compare the requirements specified in the URD against working systems.

    URD divided software into components. Workplan ordered components according to priority.

Back-end DAQ: Pre-design investigations

- Deliverables
    - Evaluation note of technologies thought to be capable of satisfying the URD
- Organisation
    - Details of each evaluation defined in work-plan
    - Organised as small working groups (max. 4 people) - one for each technology
    - Used custom-made Framemaker technical note template
- Duration
    - 5 months (Jun-Oct’96)
- Review
    - Every evaluation technical note was reviewed in the back-end DAQ meetings
    - Based on the results a single technology was selected for each area (except GUIs: Motif & Java)

Back-end DAQ: High-level design

- Deliverables

  high-level design for the component with a document containing:

  - a short textual overview of the design
  - descriptions of the interfaces to other components, sub-systems and users
  - diagrams taken from the OMT/Booch methods, produced with StP, describing the various aspects of the design

- Organisation

  Initially 5 small groups (one per “core” component)

  Groups concentrated by institute (to avoid excessive travel)

  OMT/StP training organised on CERN site

  StP repository set-up (at CERN)

- Duration

  5 months (Oct’96-Apr’97)

- Review

  Every high-level design document was reviewed in back-end DAQ meetings

  Several revisions of the documents were made as the designs evolved

  Dropped Partition Manager component (not enough information)

  Discovered Information Service (general on-line information exchange)

Back-end DAQ: Detailed design and implementation

- Deliverables
  - Unit-tested implementations of the “core” components according to the high-level design
  - User & Programmer documentation
- Organisation
  - Organised as small working groups (max. 5 people) - one for each component
  - Generally the same individuals have followed a component through design and implementation
  - Used custom-made Framemaker technical note template
  - Use StP code generation where possible (e.g. CHSM, OKS, Objectivity)
- Duration
  - on going (started Apr’97). Expect to be ready for integration with data-flow at Xmas’97.
- Review
  - Code reviews will be made (partially done for MRS, IS and run-control)

Back-end DAQ: testing

- Organisation

  Unit-tests based on use-cases identified in the high-level design documents

  Attempts to produce test-plans at the high-level design phase met with limited success

  Kept with the implementation in the SRT repository (/tests sub-directory)

- Tools

  Purify for memory leaks (Insure++ as well but less liked)

  StateMate (for run-control component finite state machine simulation)

  CHSM debug tools (for finite state machines)

  Logiscope (code coverage & metrics)

- Future

  StP/T - test-case generator tool for APIs

  Coding rule checker (off-line experience)

Summary

- Phases

  Dividing the project into several well defined phases has helped pace and organise our work

  Each phase has an obvious deliverable (i.e. document or code)

  In general, everyone know in which phase their current is defined

  The requirements phase helped enormously in defining the scope and boundaries of the project and showed differences of point of views

- Organisation

  Small is beautiful.

  Localised development greatly eases communication

  Component structure has helped to focus work

- Tools and Methods

  Adoption of the OMT method was more important that the StP CASE tool

  Method gives a common language between groups and individuals which helps dispel misunderstandings

- Future

  We have not covered all the phases of the cycle: further testing, integration, deployment, upgrades.

SW Dev. Env. history and status

- ATLAS SW Dev. Env. User Requirements Document
  - defined by an ATLAS wide group including Trigger & DAQ
  - included in ATLAS Computing Technical Proposal
  - Referenced from the ATLAS Software Process
  - possible LHC-wide project (LCB)

- Implemented in ATLAS DAQ Prototype:
  - applying simplified version of ASP
  - URD and technical note templates
  - OMT method & StP commercial case tool
  - StP customisation: code/doc generators
  - Sniff, Insure, Logiscope commercial coding/testing tools
  - Software Release Tools (SRT) for configuration management

Components and phases

Analysis and design		Delivery	
methods	CASE tools	packaging	distribution
Implementation and Integration		Verification and Validation	
general purpose libraries	languages	tracing	API test-case generator
build-tools (make)	compilers	language verifier	static analyser
style-guides	interpreters	GUI testers	code coverage
	debuggers	performance analyser	run-time error detection
Configuration Management			
defect tracking			repository
Document preparation system			
Human communication tools			
Training			
Project Management			

Training

- Need training for **all** developers

  ~25 DAQ people followed OMT/StP (we introduced the course at CERN)

  ~10 C++

  5 Objectivity

- Recognise the need for a defined training plan

  contributing to the definition of the new CERN training program

- Training must cover **all** tools and techniques used

  insist on **design** not just **programming**

- Make as much use of online tools as possible

  FAQs, news-groups, discussion lists, web tutorials, video conferencing