Software Development and Testing
Approach and Challenges
in a distributed HEP Collaboration

Doris Burckhart-Chromek
CERN PH
ATLAS TDAQ

ICALEPCS 2007 Knoxville, Tennessee
Content

- The ATLAS Experiment
- The Trigger and Data Acquisition Project
- The SDP challenge in ATLAS Online
- SDP Models
- Software Management
- Review and Inspection
- The ALICE Online SDP
- Testing
- Do’s and Don’ts
- Conclusions
The ATLAS Detector
At the Large Hadron Collider at CERN

pp collisions at 14 TeV every 25 ns: Higgs, Supersymmetry and more

Weight: 7 000 tonnes
Diameter 25 m
Length 46 m

140 million electronic channels
3 000 km of cables

Software Development and Testing: Approach and Challenges in a distributed HEF Collaboration
Doris Bürckhart Cernišek CERN PH – ICALEPCS 2007
ATLAS Collaboration

35 countries
165 institutions
2000 scientific authors
Physics data are distributed world wide
The ATLAS Experiment
Current Status

Close to completion

Move of the Endcap Toroid (280 tonnes) into the ATLAS hall, July 2007
ATLAS Trigger / DAQ Data Flow

SDX1

ATLAS detector

UX15
ATLAS Trigger / DAQ Data Flow

Event data pushed @ ≤ 100 kHz, 1600 fragments of ~ 1 kByte each
Event data pulled:
- partial events @ ≤ 100 kHz,
- full events @ ∼ 3 kHz

Event data pushed @ ≤ 100 kHz,
1600 fragments of ∼ 1 kByte each
ATLAS Trigger / DAQ Data Flow

Event data pulled:
- partial events $\leq 100$ kHz,
- full events $\leq 3$ kHz

Event data pushed $\leq 100$ kHz,
1600 fragments of $\sim 1$ kByte each
Event data pulled: partial events @ ≤ 100 kHz, full events @ ~ 3 kHz

Event data pushed @ ≤ 100 kHz, 1600 fragments of ~ 1 kByte each
ATLAS TDAQ environment

No SDP model suitable ‘as is’

- The project
  - Long start-up time - 10 years
    - Technical requirements not all known when starting
    - Subset of the system needed early on in test beams
    - Advanced system for detector commissioning
  - Long life time of the experiment – 15 years
  - 60 participating institutes located all around the world
    - Nominally 400 collaborators
  - No strong hierarchical management power
    - In a scientific collaboration we rely on participation by conviction

- The team in the scientific environment
  - Large, geographically widespread team
  - High turnover of developers
    - Short time contracts
    - Additional duties at universities and in labs
  - Predominantly physicists with some participation of computer scientists
    - Reluctant if faced with a working framework and rules to obey
    - Development habits from small projects
    - No specialized programming expertise
ATLAS TDAQ SDP

‘Light’ SDP

- Structured organization as a helpful framework
- Integration of team members
  - Introduce process gently
    - Initially introduced the SDP in a sub-group with 12 components and 10 part time developers
    - Later the other TDAQ sub-systems were gradually integrated: 160 packages, 30 core developers
    - Handle geographical distribution and turnover of team members
- Define and agree on common goals and priorities
  - \(\text{=> community feels responsible for the results}\)
  - \(\text{=> increased communication}\)
- Value and use incomplete contributions
  - Based on ‘best effort’ approach - not perfect
  - ‘We are not gurus, just concerned developers’
- Allow for iterations

Software Development and Testing: Approach and Challenges in a distributed HEP Collaboration
Doris Burchhart Chromek, CERN PH – ICALEPCS 2007
The Waterfall model

or 'linear sequential model'
The Waterfall model

or 'linear sequential model'
The Waterfall model
or ‘linear sequential model’

- The oldest and the most widely used paradigm
- Clear development phases
- Enforced disciplined approach
- Testing inherent to every phase
- Documentation driven
- Missing check points with the user during development
- Iterative cycles allowed only to each previous phase
- Maintenance paths to all phases
The Waterfall model
(or 'linear sequential model')

- The oldest and the most widely used paradigm
- Clear development phases
- Enforced disciplined approach
- Testing inherent to every phase
- Documentation driven
- Missing check points with the user during development
- Iterative cycles allowed only to each previous phase
- Maintenance paths to all phases
Other models and methods

**Spiral lifecycle model**

*Barry Boehm, 1988 “A Spiral Model of Software Development and Enhancement”*

- Each iteration is carefully planned from the start
- Risk assessment
  - Prototyping, simulation, benchmarking
  - Estimates get more realistic as work progresses, because important issues are discovered earlier
- Ability to accommodate change
- Rigid approach

**Agile methods**

- Aimed for projects with rapidly changing requirements
  - Web oriented tasks
- Rapid evolutionary delivery throughout the project lifecycle
  - Days to weeks
- Emphasis on testing
  - Tests are written before component code
- Pair programming is favored
- Used for small project teams located in one room or close by
- Little documentation
- No concept for collaboration with remote sites
ATLAS Online SDP

Based on iterative development
Iterative cycles are allowed
between each of the phases

Embedded in a
Software Management Environment

Assisted by Tools for
Process management
Programming
Release management
Configuration
SDP Organization

- Work is organized around components
  - Small groups are dedicated to each component
  - Prefer one institute per component -> clear boundaries of responsibilities
  - Look for commonalties between components - don’t duplicate functionality
  - Encourage to take / reuse as much as possible from colleagues: code, ideas, documentation, templates, examples
- Components are developed according to an agreed priority
- SDP documented on the Web allowing for change
  - Brief Check-lists and guidelines
    - For requirements, design and general documentation
    - Coding standards, automatic checking tools (C++)
    - Simple “how-to” instructions for most commonly used tools
- Documentation Templates
  - From external sources or developed within the project
  - Provides type of content & style and includes examples
  - Generic technical note, test plan & test report, user requirements, design document, users guide
- Regular reviews
Software Management Tools in ATLAS

Coherent releases allow to join the force of the project developers

- Nightly build for 2 compilers on Linux, optimized and debug versions
- Now 3-4 major releases per year
- 160 packages
- 4000 source files
- 1,000,000 lines of code
- C++, Java, Python
- Total 60Mbytes
- 2 GByte total size of nightly build
- Release nodes part of the build
Software Management Tools in ATLAS

Coherent releases allow to join the force of the project developers

- Nightly build for 2 compiler optimized and debug versions
- Now 3-4 major releases per year
- 160 packages
- 4000 source files
- 1 000 000 lines of code
- C++, Java, Python
- total 60Mbytes
- 2 GByte total size of nightly build
- Release nodes part of the build

Software Development and Testing - Approach and Challenges in a distributed HEF Collaboration
Doris Burekbert Chromek, CERN PH – ICALEPCS 2007
Software Management Tools in ATLAS

Coherent releases allow to join the force of the project developers

- Nightly build for 2 compilers on linux, optimized and debug versions
- Now 3-4 major releases per year
- 160 packages
- 4000 source files
- 1 000 000 lines of code
- C++, Java, Python
- total 60Mbytes
- 2 GByte total size of nightly build
- Release nodes part of the build
Software Management Tools in ATLAS

Coherent releases allow to join the force of the project developers

- Nightly builds optimized and tested
- Now 3-4 major releases
- 160 packages
- 4000 source files
- 1,000,000 lines of code
- C++, Java, Python
- total 60Mbyte
- 2 GByte total
- Release notes

Software Development and Testing: Approach and Challenges in a distributed HEP Collaboration
Doris Erecbhart Chomelnik CERN PH – ICAL EPCS 2007
Software Management Tools in ATLAS

Coherent releases allow to join the force of the project developers

- Nightly build for 2 compilers on linux, optimized and debug versions
- Now 3-4 major releases per year
- 160 packages
- 4000 source files
- 1,000,000 lines of code
- C++, Java, Python
- Total 60Mbytes
- 2 GByte total size of nightly build
- Release nodes part of the build

- Code versioning system CVS
- Configuration management system CMT
- Custom made scripts
- Build process takes 3-4 hours thanks to serialization and parallelism
- Presentation of build status on web pages
- Automatic information of component failures to the corresponding developer
- RPM for package building
- APT for package distribution
- VALGRIND as preferred memory debugger and performance profiler
- doxygen for automatic documentation generation of the application interfaces
Informal Review

Informal Review: Clarification and accept/reject decision

- Presentation of each component to the group in each development phase
- Discussion and coordination with other components

- In ATLAS TDAQ SDP applied from the start of the project
  - document preparation and monthly open meetings
  - present status, results, proposals
  - inform colleagues - receive feedback
  - suggestions -> enhancements -> acceptance

- Results:
  - Coherent set of end-product components
  - Increased communication
  - Created the basis for a working culture

- Drawback:
  - Lack of time of reviewers
  - No code review
The Power of Software Inspection

Formal Inspection:
Quality Improvement Process to the software project

Defect Detection
- Documents are checked for cleanliness and consistency against rules

Defect Prevention
- Participants
  - learn from defects found
  - and suggest improvements

On the Job Training & Integration
- Peers
  - get familiar with standards and rules
  - apply creativity
  - learn to accept criticism
  - learn to trust colleagues
  - get integrated into the group & project
Inspection Process Map

Based on Tom Gilb’s Inspection method

Sources
- Rules
- Checklists
- Input Product

Planning & Entry
- Kick-off Meeting
- Checking
- Logging Meeting
- Brainstorming
- Edit
- Follow-up
- Exit

Inspection Plan
- Issue log tables
- Action Lists

Inspection Team:
- Inspection Leader
- Authors
- Inspectors

Change Requests

Vehicle to build a project culture

Software Development and Testing. Approach and Challenges in a distributed HEP Collaboration
Doris Eurekhart-Chromek, CERN PH – ICALEPCS 2007
Inspection & Review: Experience

Requirements
- Highest payoff

Design
- Use walkthrough methods

Code
- Inspect samples
- Need a good set of rules
- Use automatic checking tools

General
- Method had to be adapted to the Environment
- Gain in quality and experience
- Appreciated by authors and peers
- Inspections prepare the ground and stabilize SDP
- Help for team building in a distributed environment
- Helped for breakthrough in politically prominent packages
Light form of inspection

- Can be applied when
  - working habits are assimilated by the team
  - criticism is accepted
- Retains primary benefits of inspection
- Meetings are replaced by the use of electronic communication tools
- Less time consuming for reviewers
- Less organizational overhead

- Not with reviewers which are external to the project
- Not for politically prominent components
The SDP in ALICE Online

- ALICE detector at the LHC is smaller than ATLAS
- TDAQ has high requirements on the performance of physics data storage
- Development team
  - Team started with 4 very experience developers
  - Familiarity of team members since a decade and a common history
  - Long term experience in development and support
  - Newcomers could be integrated stepwise (up to 10)
  - Based at CERN
  - Immediate task to provide a TDAQ system for related experiments
- Approach
  - Short software life cycle because of ongoing experiments
  - Common understanding, conventions, development techniques were a given
  - XP like programming
  - But unlike XP: Thorough user documentation from the start and maintained
    - -> allowed to discover discrepancies
  - Emphasis on Testing in test beds and as dedicated test in collaboration with the CERN computer center
  - Thorough requirements documents and reviews in fields which were new to the team

12000 lines of code
O(10) packages
CVS, RPM, YUM
5-10 releases per year
The SDP in ALICE Online

- ALICE detector at the LHC is smaller than ATLAS
- TDAQ has high requirements on the performance of physics data storage
- Development team
  - Team started with 4 very experience developers
  - Familiarity of team members since a decade and a common history
  - Long term experience in development and support
  - Newcomers could be integrated stepwise (up to 10)
  - Based at CERN
  - Immediate task to provide a TDAQ system for related experiments
- Approach
  - Short software life cycle because of ongoing experiments
  - Common understanding, conventions, development techniques were a given
  - XP like programming
  - But unlike XP: Thorough user documentation from the start and maintained
    - -> allowed to discover discrepancies
  - Emphasis on Testing in test beds and as dedicated test in collaboration with the CERN computer center
  - Thorough requirements documents and reviews in fields which were new to the team

Success: approach matched the team, size and conditions
Testing in the Atlas online SDP

- **Component testing with test plan**
- **Nightly check targets**
  - for each component
  - for the integrated system
- **Release testing**
  - performed by developers and testers
  - starts with infrastructure testing and then includes other sub-systems and application testing
  - on desktop PC, in labs
  - validated on a larger farm
  - patches are applied to the release if necessary
- **Integration with external software**
- **Regular field tests on smaller scale** (<100PC's)
- **Large scale & performance tests** of the integrated system
- **Deployment in commissioning tests**
Test Organization

- Chop tests into small units
- Avoid personal test-ware of developer
- Provide test programs which can serve as examples for users
- Standardize on command, output and exit codes
- Make them part of the software repository
  -> follow the evolution of the component
- Allow for flexibility and test boundaries and critical areas
- Use testing tools for code coverage and Memory leak checking
- Use same basic test software for ALL testing aspects
- Document them
- Establish use cases
- Identify critical areas, test boundaries
- Prepare scripts or utilities to group and run them in a selective manner
- Involve non-authors for testing

*In Industry: specialized department for Testing; 20% of manpower per project*
Large Scale tests

• **Study functionality on large computing farms**
• **Concentrate on operational aspects of the system**
  • Process control and state machine operation
  • Configuration
  • Concurrent database access (25,000 simultaneous clients)
  • Information exchange (50,000 objects per second)
  • Process communication
  • Granularity of sub-farms for physics data processing
  • Granularity of communication and database server layers
• **Rare events occur more frequently, become reproducible**
• **Trend analysis from results versus number of nodes**

CERN computing center &
Westgrid cluster in Canada
1-4 weeks
Large Scale tests

- Study functionality on large computing farms
- Concentrate on operational aspects of the system
  - Process control and state machine operation
  - Configuration
  - Concurrent database access (25,000 simultaneous clients)
  - Information exchange (50,000 objects per second)
  - Process communication
  - Granularity of sub-farms for physics data processing
  - Granularity of communication and database server layers
- Rare events occur more frequently, become reproducible
- Trend analysis from results versus number of nodes

CERN computing center & Westgrid cluster in Canada
1-4 weeks

<table>
<thead>
<tr>
<th>Farm size (nodes)</th>
<th>100</th>
<th>220</th>
<th>220</th>
<th>230</th>
<th>300/800</th>
<th>700</th>
<th>1000</th>
</tr>
</thead>
</table>

Software Development and Testing: Approach and Challenges in a distributed HEF Collaboration
Doris Buechthart Chromek: CERN PH – ICALEPCS 2007
Experience and examples

- Learning process on how to conduct the tests
- Number of problems grows significantly with scale (> 500 nodes)
- Experienced the need for
  - Fault tolerance and stability
    - Recovery from system failure is too costly in data taking time if having to stop and restart
  - System monitoring and automatic repair
  - Farm management tools and testing tools

- Example:
  Database access is sensitive to scale efficient access methods should be used
  - Use of distributed remote configuration database servers
  - Optimize access mode at the application level
Experience and examples

- Learning process on how to conduct the tests
- Number of problems grows significantly with scale (> 500 nodes)
- Experienced the need for
  - Fault tolerance and stability
    - Recovery from system failure is too costly in data taking time if having to stop and restart
  - System monitoring and automatic repair
  - Farm management tools and testing tools

- Example:
  Database access is sensitive to scale efficient access methods should be used
  - Use of distributed remote configuration database servers
  - Optimize access mode at the application level
The Vertical Slice Test Bed

- 80 PCs in the experimental area
  - including final physics data readout hardware,
  - permanently available
- Test conditions close to final ATLAS
  - Network separated form the CERN network
  - Computing hardware
  - Logistics
- Concentrate on the data taking phase
  - Stability of a data taking run
  - Performance
  - Physics event data transport
- Validate simulation of components and subsequently model the full size ATLAS
- Validate technology and implementation choices
Deployment and Status

- **Technical runs and detector commissioning**
  - Organization towards final ATLAS
  - Conducted on the final system
  - Controlled from the ATLAS control room
  - Run for 1-2 weeks, every 1-2 months
  - Work plan, preparation
  - Run coordinator, shifts, e-log book
  - On the job training for shifters to become experts
  - Important activities for team building
  - Efficient occasion to get user feedback and suggestions for enhancements
  - Attractive activity for external collaborators to participate
  - Gives a momentum to people who have worked since many years on the project

**Current Status**
- All TDAQ sub-systems integrated
- Readout of all ATLAS detectors integrated
- A lot still to be done
Deployment and Status

- Technical issues in detector:
  - Cosmic muons
  - Failed attempts to try tdaq-01
  - SW installation: tried to upgrade firmware earlier today
  - Cosmic muon run failed

- Barbell Data completely corrupted
- Dock signal not transmitted between CTP & LTP

- Attractive activity for external collaborators to participate
- Gives a momentum to people who have worked since many years on the project

Software Development and Testing: Approach and Challenges in a distributed HEP Collaboration
Doris Bueckhart-Chromek CERN PH – ICALEPCS 2007
Deployment and Status

- **Technical runs and detector commissioning**
  - Organization towards final ATLAS
  - Conducted on the final system
  - Controlled from the ATLAS control room
  - Run for 1-2 weeks, every 1-2 months
  - Work plan, preparation
  - Run coordinator, shifts, e-log book
  - On the job training for shifters to become experts
  - Important activities for team building
  - Efficient occasion to get user feedback and suggestions for enhancements
  - Attractive activity for external collaborators to participate
  - Gives a momentum to people who have worked since many years on the project
Do’s and Don'ts

- Choose an SDP model which matches most closely
  - project size and life time
  - experience of team members and their geographical location
- **Build change management into the process**
- Give importance to **building up a project culture**
  - Invest in people
  - \( \rightarrow \) many benefits are implicit and come automatically
- **Write and review requirements thoroughly**
- **Make testing an integral part of the SDP as early as possible**
  - nightly builds and check targets
- Use code management and configuration management system from the start
- Use project management tools and collaborative tools
- Invest in explicit and on the job **training**
- Seek **management support** and a **driving person**
Conclusions

The ATLAS TDAQ project has benefited in many aspects from adapting a flexible SDP framework.
Current deployment in detector commissioning activities demonstrates the success of the team and of the approach.

Thanks to

- the ATLAS TDAQ project members and its management
- Pierre Vande Vyvre from the ALICE collaboration
- Bob Jones who has started the SDP in ATLAS TDAQ