Applications Session Summary

John Harvey (chair)
Torre Wenaus (project leader)

Marco Cattaneo (recorder)

Session Contents

- Torre Wenaus 's vision for the Applications subproject
  - Torre started officially March 1st – now based at CERN
- Reports received from the first 3 RTAGs
  - Process for managing LCG software (final report)
  - Math library review (interim report)
  - Data persistency (interim report)
- More than 1 hour in total devoted to discussion

Status

- The LCG mechanisms are in use
- The SC2 created first 3 RTAGs in January
  - All 3 RTAGs are Applications related
  - The first RTAG is close to submitting its final report
  - The PEB can expect to receive its first project proposal soon
- In meantime extensive preparatory work done by PEB, and by Torre in particular
  - Input collected from all parties in discussions
  - Suggestions for work programme were presented in form of 'candidate RTAGs'
  - Possible approach to organisation and management of execution of the project presented
  - Aim was to provoke meaningful discussion – many important issues flagged to be followed up

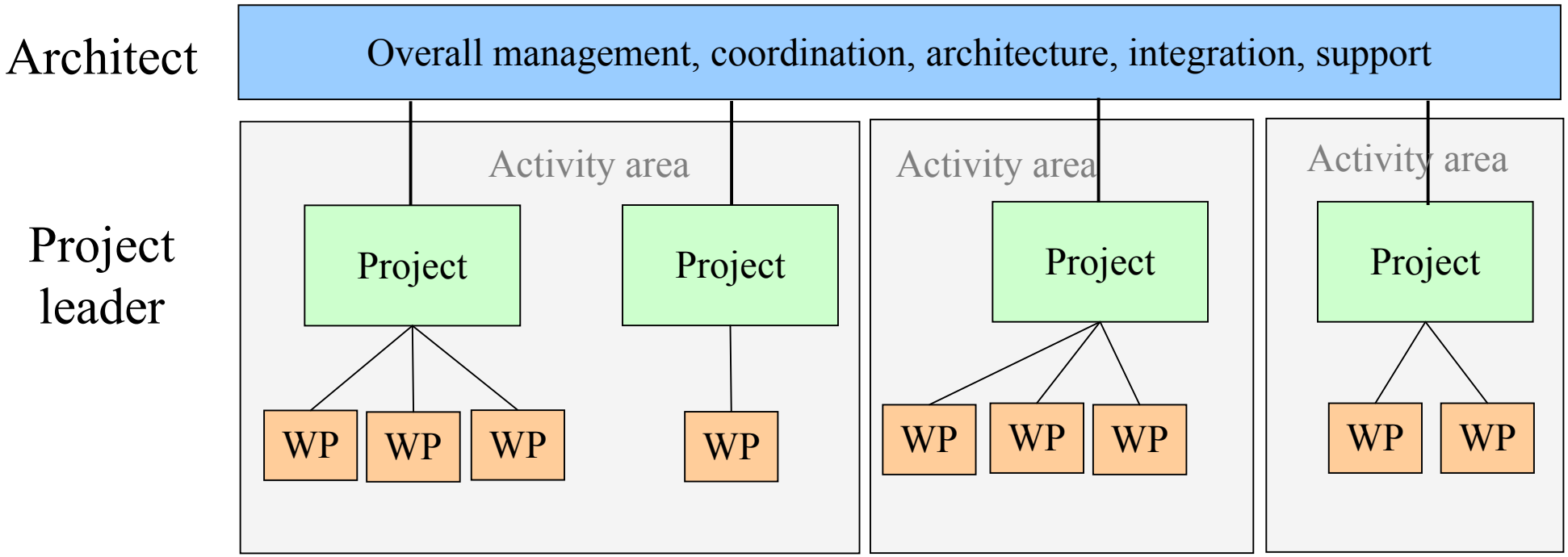
Scope of Applications Area

1. Application software infrastructure
2. Common frameworks for simulation and analysis
3. Support for physics applications
4. Grid interface and integration
5. Physics data management

Issues:

- 'How will applications area cooperate with other areas?'
- 'Not feasible to have a single LCG architect to cover all areas.'
- Need mechanisms to bring coherence to the project

Possible Organisation of activities



Example:

Activity area: Physics data management
Possible projects: Hybrid event store, Conditions DB, …
Work Packages: Component breakdown and work plan lead to Work Package definitions. ~1-3 FTEs per WP

Issues related to organisation

- There is a role for a Chief Architect in overseeing a coherent architecture for Applications software and infrastructure
  - This is Torre's role
- 'We need people who can speak for the experiments – should be formalised in organisation of the project'
  - Experiment architects have key role in getting project deliverables integrate into experiment software systems
  - They must be involved directly in execution of the project
- Mechanisms need to be found to allow for this e.g.
  - Regular meetings (monthly?) of the Applications subproject that must be attended by architects at which decisions are taken
  - Difficult decisions are flagged and referred to a separate meeting attended by experiment architects and coordinators
  - Ultimate responsibility and authority is with the Chief Architect

Candidate RTAGs by activity area

- Application software infrastructure
  - Software process; math libraries; C++ class libraries; software testing; software distribution; OO language usage; benchmarking suite
- Common frameworks for simulation and analysis
  - Simulation tools; detector description, model; interactive frameworks; statistical analysis; visualization
- Support for physics applications
  - Physics packages; data dictionary; framework services; event processing framework
- Grid interface and integration
  - Distributed analysis; distributed production; online notebooks
- Physics data management
  - Persistency framework; conditions database; small scale persistency

Candidate RTAG timeline

	Months	Merge?	02Q1	02Q2	02Q3	02Q4	03Q1	03Q2	03Q3	03Q4
Simulation tools	1			X						
Detector description & model	2			X						
Conditions database	1				X					
Data dictionary	2			X						
Interactive framew orks	2			X						
Statistical analysis	1				X					
Detector & event visualization	2					X				
Physics packages	2				X					
Framew ork services	2				X					
C++ class libraries	2				X					
Event processing framew ork										X
Distributed analysis interfaces	2						X			
Distributed production systems	2					X				
Small scale persistency	1							X		
Softw are testing	1				X					
Softw are distribution	1					X				
OO language usage	2							X		
LCG benchmarking suite	1						X			
Online notebooks	2								X	

Issues related to partitioning the work

- 'How do you go from present to future without dismantling existing projects?' (Rene Brun)
- 'Have to be careful that we don't partition into too small chunks and lose coherence of overall software' (David Stickland)
- We are not starting afresh, we have a good knowledge of what the broad categories are going to be (Norman McCubbin)
- Experiment architectures help to ensure coherency. (Vincenzo Innocenti)

Coherent Architecture

- Applications common projects must follow a coherent overall architecture
- The software needs to be broken down into manageable pieces i.e. down to the component level
- Component-based, but not a bag of disjoint components
  - components designed for interoperability through clean interfaces
  - Does not preclude a common implementation foundation, such as ROOT, for different components
    - The 'contract' in the architecture is to respect the interfaces
    - No hidden communication among components
  - Starting point is existing products, not a clean slate

Distributed Character of Components

- Concern that distributed character of components was not being addressed
- Distributed analysis
- Distributed production
- Software distribution
  - Should use the grid
- Interactive frameworks
  - Grid-aware environment; 'transparent' access to grid-enabled tools and services
- Persistency framework
  - Naming based on logical filenames
  - Replica catalog and management
- Conditions database
  - Inherently distributed (but configurable for local use)

Approach to making workplan

- "Develop a global workplan from which the RTAGs can be derived"
- Considerations for the workplan:
  - Experiment need and priority
  - Is it suitable for a common project
  - Is it a key component of the architecture e.g. object dictionary
  - Timing: when will the conditions be right to initiate a common project
    - Do established solutions exist in the experiments
    - Are they open to review or are they entrenched
  - Availability of resources
  - Allocation of effort
    - Is there existing effort which would be better spent doing something else
  - Availability, maturity of associated third party software
    - E.g. grid software
- Pragmatism and seizing opportunity. A workplan derived from a grand design does not fit the reality of this project

Global Workplan – 1st priority level

Initiate: First half 2002

1. Establish process and infrastructure
   - Nicely covered by software process RTAG
2. Address core areas essential to building a coherent architecture
   - Object dictionary – essential piece
   - Persistency - strategic
   - Interactive frameworks - also driven by assigning personnel optimally
3. Address priority common project opportunities
   - Driven by a combination of experiment need, appropriateness to common project, and 'the right moment' (existing but not entrenched solutions in some experiments)
     - Detector description and geometry model
   - Driven by need and available manpower
     - Simulation tools

Global Workplan – 2nd priority level

Initiate: Second half 2002

- Build outward from the core top-priority components
  - Conditions database
  - Statistical analysis
  - Framework services, class libraries
- Address common project areas of less immediate priority
  - Math libraries
  - Physics packages (scope?)
- Extend and elaborate the support infrastructure
  - Software testing and distribution

Global Workplan – 3rd priority level

Initiate: First half 2003

- The core components have been addressed, architecture and component breakdown laid out, work begun. Grid products have had another year to develop and mature. Now explicitly address physics applications integration into the grid applications layer.
  - Distributed production systems. End-to-end grid application/framework for production.
  - Distributed analysis interfaces. Grid-aware analysis environment and grid-enabled tools.
- Some common software components are now available. Build on them.
  - Lightweight persistency, based on persistency framework
  - Release LCG benchmarking suite

Global Workplan – 4th priority level

Initiate: Second half 2003 and later

- Longer term items waiting for their moment
  - 'Hard' ones, perhaps made easier by a growing common software architecture
    - Event processing framework
  - Address evolution of how we write software
    - OO language usage
  - Longer term needs; capabilities emerging from R&D (more speculative)
    - Advanced grid tools, online notebooks, …

Resources

- Original manpower-required estimates from Sep 2001 proposal
  - To be completely revisited, but gives some idea of numbers and foreseen distribution
- Total: 36 FTEs of which 23 are new LCG hires
  - Application software infrastructure - 5 FTEs
  - Common frameworks for simulation & analysis - 13 FTEs
  - Support for physics applications – 9 FTEs
  - Physics data management - 9 FTEs
- 'Mountain of manpower will be with us very soon and will leave in 3 years'
  - Tentative project assignments according to abilities
  - Setting up tools and infrastructure
- 'Some new people could go to experiment to get up to speed, whilst more experienced could go from experiments to project'
- 'How does support issue effect project'
  - Support needed for lifetime of the software
  - Long term commitments to support the software needed

Process RTAG

- Mandated to define a process for managing LCG software
- Development practices proposed by RTAG should be followed by all LCG projects
  - tools, standards and procedures must be centrally installed, maintained and supported
- LCG should define one common software development process based on current best practices e.g. XP, RUP, USDP
  - Architecture-centric
  - Iterative and incremental approach to software development
    - Release early, release often
    - Early exposure to users, regular feedback
  - Use-case driven ("Let user feedback drive the development")
- Tools and procedures proposed to support such a process

Process RTAG : Choice of Tool

- Formal approach is to define requirements and evaluate wide range
  - This will take a lot of time and effort
  - Simple approach – see what is used today and choose best
- Tools exist that have been developed and are used by the various teams (e.g CMT and SCRAM)
  - Converging on a single tool will ease maintenance and simplify sharing of software
  - Changing will have an impact, not a rapid process
  - In short term need to understand the work models supported, the commonalities and differences
- Where one tool is chosen by LCG, look to provide an interface to the other to cover short term needs

Process RTAG : Recommendations

- 'Release early, release often' implies
  - major release 2-3 times per year
  - Development release every 2-3 weeks
  - Automated nightly builds, regression tests, benchmarks
- Test and quality assurance
- Support of external software
  - installation and build up of local expertise
- Effort needed for filling support roles
  - Librarian
  - Release manager
  - Toolsmith
  - Quality assurance
  - Technical writer

Math Library Review RTAG

- Many different libraries in use
  - General purpose (NAG-C, GSL, ..)
  - HEP-specific (CLHEP, ZOOM, ROOT)
  - Modern libraries dealing with matrices and vectors (Blitz++, Boost..)
- Financial considerations
  - NAG-C 300 kCHF/yr initially dropping to 100 kCHF/yr after 4-5 years
- Comparative evaluation of NAG-C and GSL is difficult and time-consuming
  - Collect information on what is used/needed
  - Evaluation of functionality and performance very important
- HEP-specific libraries expected to evolve to meet future needs
- This was an interim report – work is continuing

Data Persistency RTAG

- Mandated to write the product specification for the Persistency Framework for Physics Applications at LHC
- *Collaborative, friendly atmosphere" "Real effort to define a common product"*
- Focused on the architecture of a persistence management service
- Aim is to define components and their interactions in terms of abstract interfaces that any implementation must respect
- RTAG's highest priority is to provide the foundation for a near-term common project reasonably matched to current capabilities of ROOT, with a relational layer above it
- Optimistic about prospects to accomplish this—significant progress to date
- Additional work (further RTAGs) in other areas will almost certainly be necessary— recommendations will be made

Next Steps

- Send outcome of RTAGs 1, 2 and 3 to PEB
  - Complete the cycle RTAG – SC2 – PEB - WP
- Experiments carry away list of candidate RTAGs and discuss
- Proposals brought to SC2
- Launch next group of RTAGs
- Follow up on organisational issues
- Embed new effort as it arrives

Supplementary slides

Candidate RTAGs (1)

Simulation tools	Non-physics activity; ask SC2
Detector description, model	Description tools, geometry model
Conditions database	If necessary after existing RTAG
Data dictionary	Key need for common service
Interactive frameworks	What do we want, have, need
Statistical analysis	Tools, interfaces, integration
Visualization	Tools, interfaces, integration
Physics packages	Important area but scope unclear
Framework services	If common framework is too optimistic…
C++ class libraries	Standard foundation libraries

Candidate RTAGs (2)

Event processing framework	Hard, long term
Distributed analysis	Application layer over grid
Distributed production	Application layer over grid
Small scale persistency	Simple persistency tools
Software testing	May be covered by process RTAG
Software distribution	From central 'Program Library' to convenient broad distribution
OO language usage	C++, Java (..?) roles in the future
Benchmarking suite	Comprehensive suite for LCG software
Online notebooks	Long term; low priority

Post-RTAG Participation of Architects – Draft Proposal (1)

- Monthly open meeting (expanded weekly meeting)
  - Accumulated issues to be taken up with architects
  - Architects in attendance; coordinators invited
- Information has gone out beforehand, so architects are 'primed'
- Meeting is informational, and decision-making (for the easier decisions)
  - An issue is either
    - Resolved (the easy ones)
    - Flagged for addressing in the *'architects committee'*

Post-RTAG Participation of Architects – Draft Proposal (2)

- Architects committee:
  - Members: experiment architects + applications manager (chair)
  - Invited: computing coordinators, LCG project manager and CTO
  - Others invited at discretion of members
    - e.g. project leader of project at issue
- Meets shortly after the open meeting (also bi-weekly?)
- Decides the difficult issues
  - Most of the time, committee will converge on a decision
  - If not, try harder
  - If still not, applications manager takes decision
    - Such decisions can be accepted or challenged
- Challenged decisions go to full PEB, then if necessary to SC2
  - PEB role of raising issues to be taken up by SC2
  - We all abide happily by an SC2 decision
- Committee meetings also cover general current issues and exchange of views
- Committee decisions, actions documented in public minutes

Distributed Character of Components (1)

- Persistency framework
  - Naming based on logical filenames
  - Replica catalog and management
  - Cost estimators; policy modules
- Conditions database
  - Inherently distributed (but configurable for local use)
- Interactive frameworks
  - Grid-aware environment; 'transparent' access to grid-enabled tools and services
- Statistical analysis, visualization
  - Integral parts of distributed analysis environment
- Framework services
  - Grid-aware message and error reporting, error handling, grid-related framework services

Distributed Character of Components (2)

- Event processing framework
  - Cf. framework services, persistency framework, interactive frameworks
- Distributed analysis
- Distributed production
- Software distribution
  - Should use the grid
- OO language usage
  - Distributed computing considerations
- Online notebook
  - Grid-aware tool