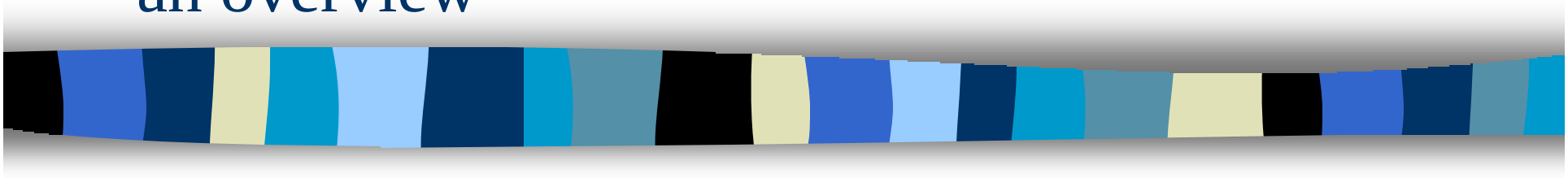
Software Process in Geant4

an overview



Gabriele Cosmo

CERN IT/API-SI

Gabriele.Cosmo@cern.ch

Outline

- Overview on Software Processes
- The area of application
- Life-cycle processes in Geant4
- Assessment model
- Software Process Improvement
- Future evolutions

Definitions...

- Software Process
  - A set of interrelated activities, which transform inputs into outputs (*ISO 12207/8402*)
    - used by an organisation or project to plan, manage, execute, monitor, control and improve any software related activity
  - Life-cycle processes are structured in *dimensions*:
    - Primary processes
      - includes all major functions of software development
    - Supporting processes
      - for supporting other processes with a purpose
    - Organisational processes
      - for corporate level management and improvement

Process Architecture

Customer-Supplier

- CUS.1 Acquisition
  - CUS.1.1 Acquisition Preparation
  - CUS.1.2 Supplier Selection
  - CUS.1.3 Supplier Monitoring
  - CUS.1.4 Customer Acceptance
- CUS.2 Supply
- CUS.3 Requirements Elicitation (*)
- CUS.4 Operation
  - CUS.4.1 Operational Use
  - CUS.4.2 Customer Support (*)

Engineering

- ENG.1 Development
  - ENG.1.1 System Requirements Analysis & Desi
  - ENG.1.2 Software Requirements Analysis
  - ENG.1.3 Software Design (*)
  - ENG.1.4 Software Construction (*)
  - ENG.1.5 Software Integration (*)
  - ENG.1.6 Software Testing (*)
  - ENG.1.7 System Integration & Testing (*)
- ENG.2 System & Software Maintenance (*)

Support

- SUP.1 Documentation (*)
- SUP.2 Configuration Management (*)
- SUP.3 Quality Assurance
- SUP.4 Verification
- SUP.5 Validation
- SUP.6 Joint Reviews
- SUP.7 Audit
- SUP.8 Problem Resolution (*)

Management

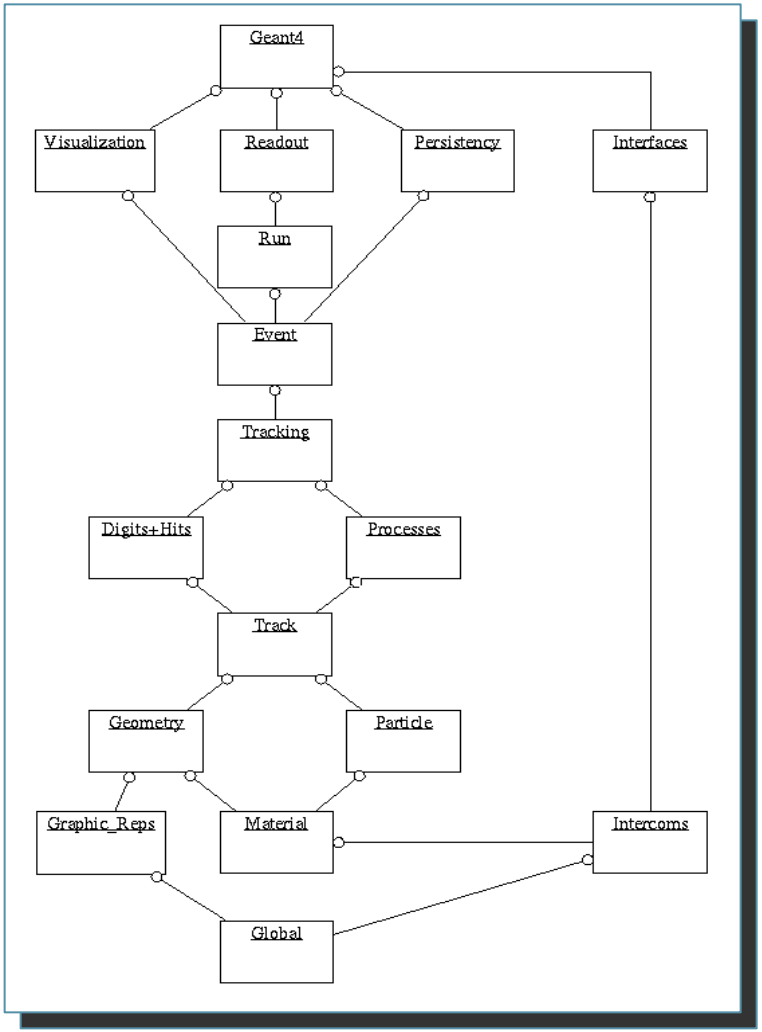
- MAN.1 Management
- MAN.2 Project Management
- MAN.3 Quality Management
- MAN.4 Risk Management

Organisation

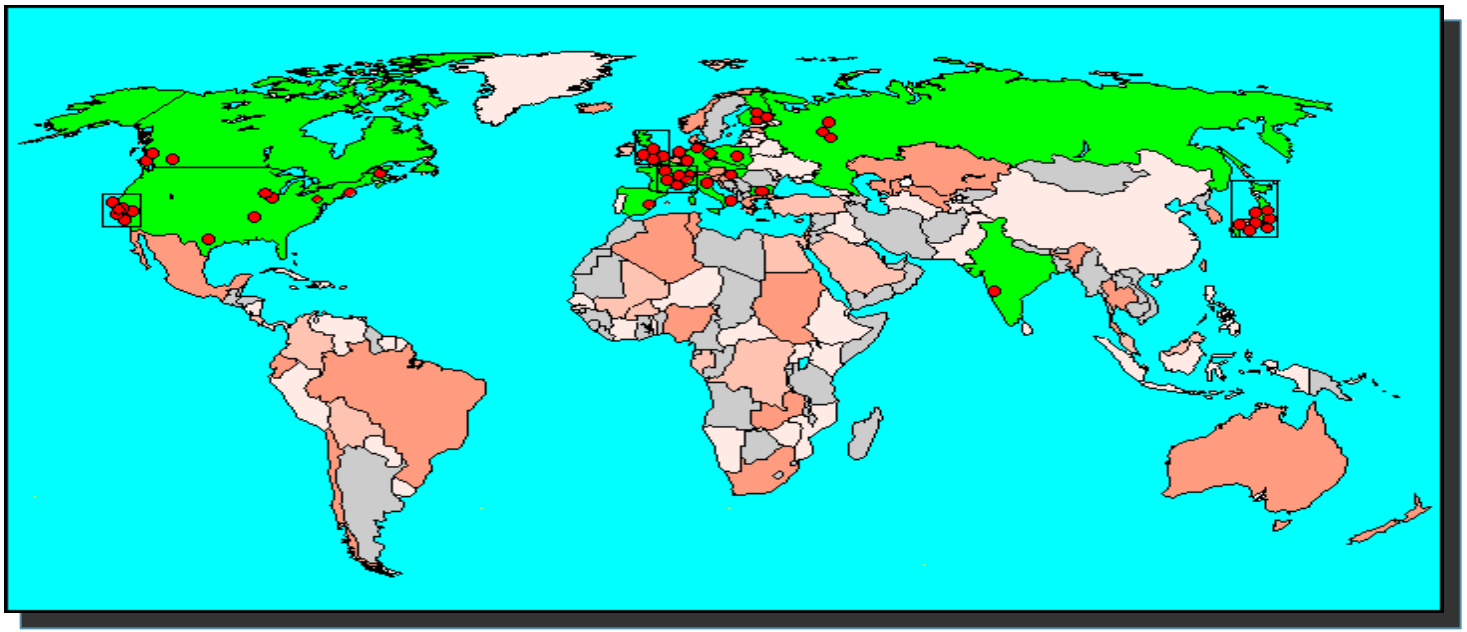
- ORG.1 Organisational Alignment
- ORG.2 Improvement
  - ORG.2.1 Process Establishment
  - ORG.2.2 Process Assessment (*)
  - ORG.2.3 Process Improvement (*)
- ORG.3 Human Resource Management
- ORG.4 Infrastructure
- ORG.5 Measurement
- ORG.6 Reuse

The area of application: Geant4

- More than 1200 classes distributed in 17 Categories
  - software components in the Booch terminology
  - complex Categories organised in a hierarchical structure
- Decomposition to domain Categories derived from the design Category diagram
  - one development team associated to one Category domain



The area of application: Geant4



- Development teams distributed world-wide
  - domain decomposition <> geographical location of teams
  - centralized coordination of domain Categories
  - local coordination of each Working Group
    - assignment of responsibilities and support
  - distributed resources and funds in a *dynamic* environment
- Coordination for a coherent development
  - computing environment, methods and tools

Requirements Elicitation

- General User Requirements (UR) collected during the R&D phase of the project (RD44)
  - GEANT3 user community involved
  - URD generated according to the ESA PSS-05 software engineering standard
  - regular update and versioning of the URD along the development process
- Change-management based on *CVS*
  - general URD currently under revision
  - maintenance and tracking of specific detailed URDs under responsibility of WG coordinators
- New requirements approval: by the TSB
  - ongoing process improvement
- Example: *physics* - see next talk

Software Design

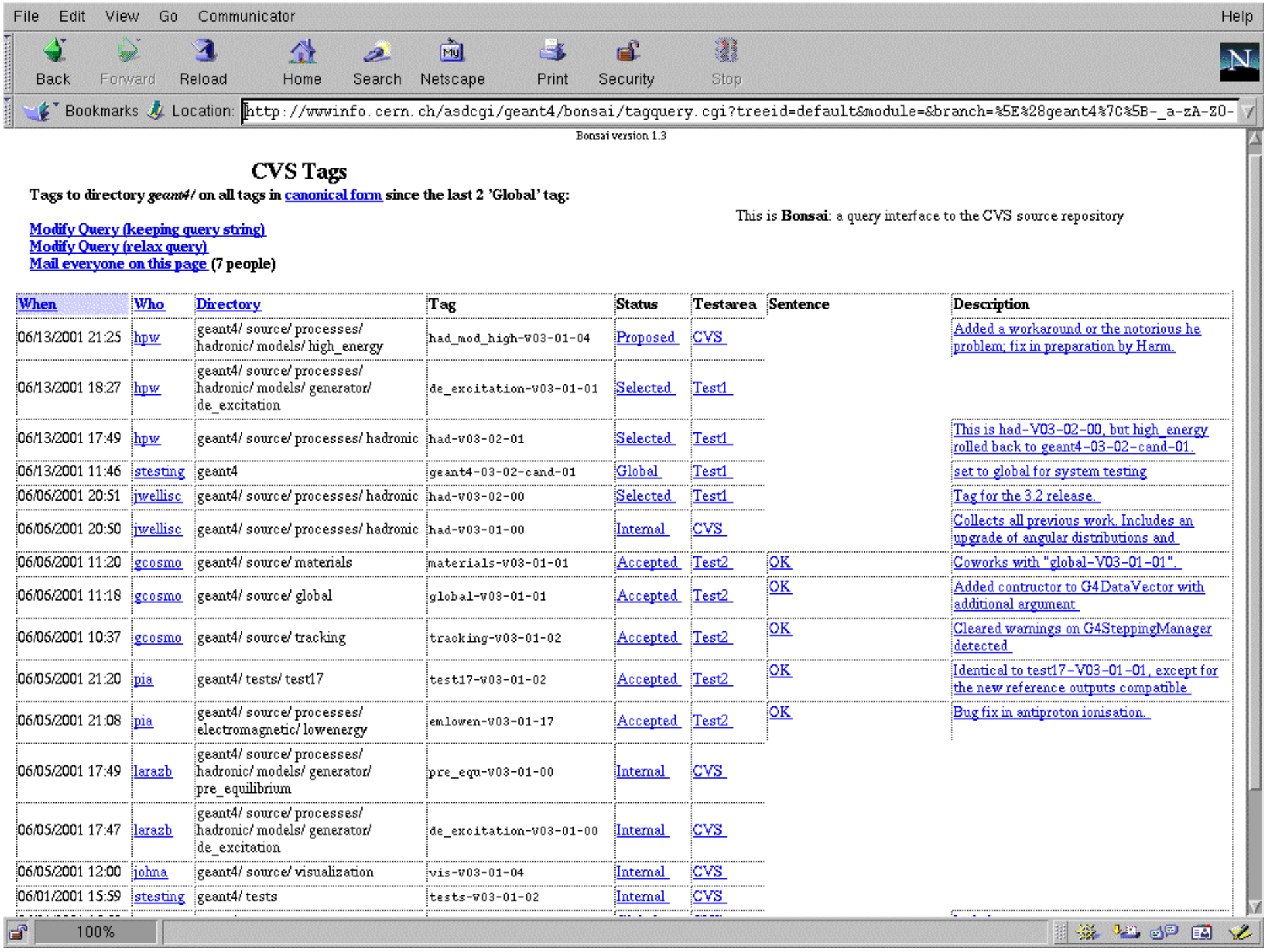
- Adoption of the *Booch* methodology for OOAD since the R&D project start
  - chosen after deep evaluation of the existing methodologies (‘94)
  - tailored to project specific needs
  - supported by CASE tools (*Rational-Rose*)
  - UML notation adopted for design documents
    - Category diagrams, Class diagrams, Scenario diagrams, Class specifications
    - ongoing process improvement
- Software development structured in *macro* and *micro* processes showed very effective
  - *iterative & incremental* approach (*spiral* model)
  - loose domain coupling led to efficient WG structure
- Example: *physics* - see next talk

Software Construction

- Software packaging reflects the domain decomposition in Categories
  - Packaging of Categories and sub-Categories in relation to definition of abstract and concrete interfaces (*frameworks*)
    - Provide a set of services in a *re-usable* way
    - Software *toolkit* approach
    - Example: *interactivity* - see next talk
- Essential and flexible guidelines for coding
- Code filtering with specialised tools
  - *Code Wizard*
  - both in the global and unit context
    - tool accessible from Web

System Testing

- Activity deployed to a specialised team (STT)
  - based on defined procedures
    - *CVS* tagging policy
    - automated through Web tools and scripts
      - Bonsai, *LXR*, *Tinderbox*
      - ongoing process improvement
  - test applications used also for system integration
    - run & tested on every supported platform/compiler
    - ongoing process improvement
  - user example applications used for acceptance
- Category tags submitted to testing in sequence according to the dependency flow dictated by the design category diagram
- Close collaboration with the release manager



CVS Tags

Tags to directory *geant4/* on all tags in canonical form since the last 2 'Global' tag:

This is **Bonsai**: a query interface to the CVS source repository

Modify Query (keeping query string)
Modify Query (relax query)
Mail everyone on this page (7 people)

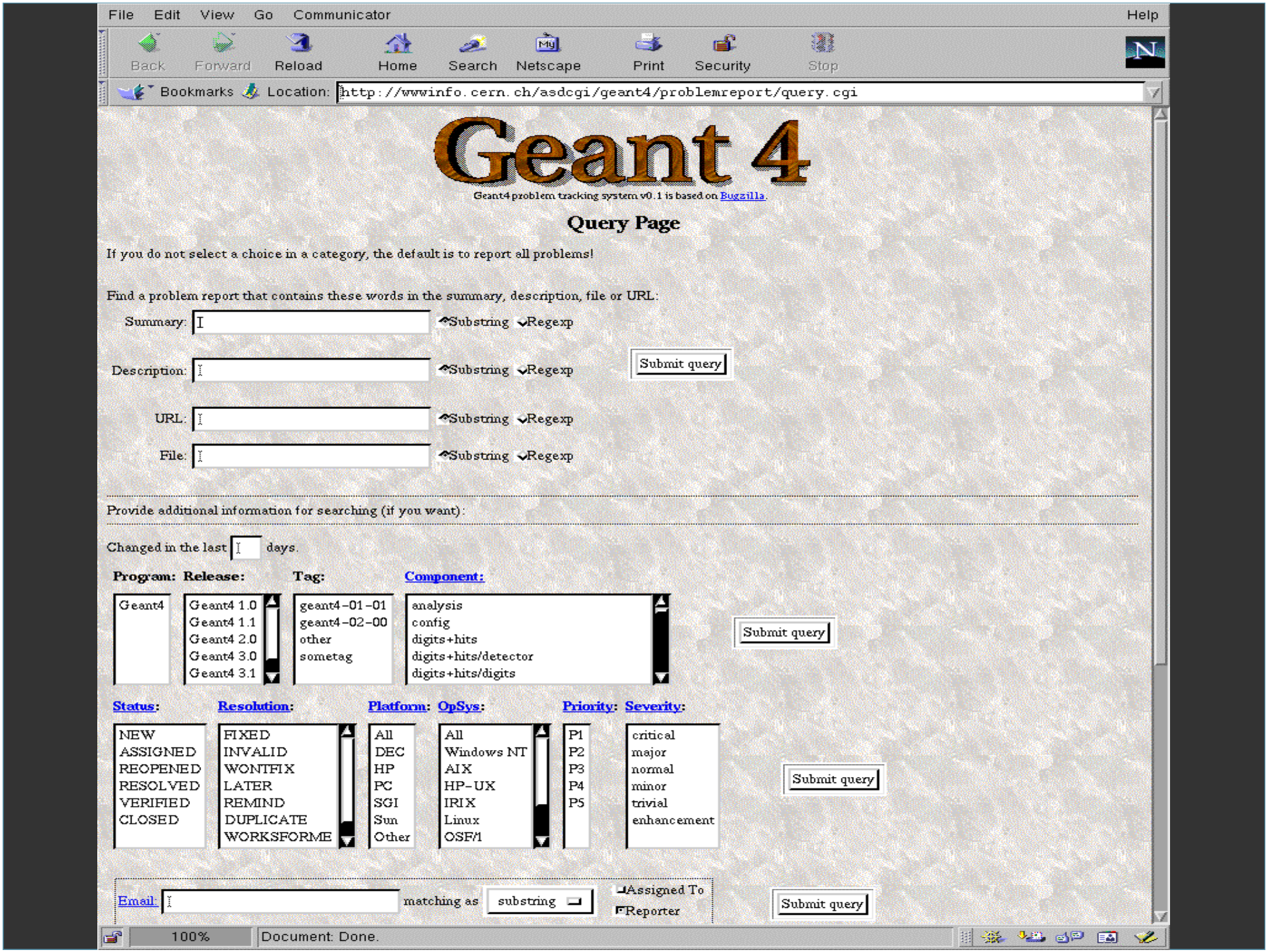
When	Who	Directory	Tag	Status	Testarea	Sentence	Description
06/13/2001 21:25	hpw	geant4/ source/ processes/ hadronic/ models/ high_energy	had_mod_high-V03-01-04	Proposed	CVS		Added a workaround or the notorious he problem; fix in preparation by Harm.
06/13/2001 18:27	hpw	geant4/ source/ processes/ hadronic/ models/ generator/ de_excitation	de_excitation-V03-01-01	Selected	Test1		
06/13/2001 17:49	hpw	geant4/ source/ processes/ hadronic	had-V03-02-01	Selected	Test1		This is had-V03-02-00, but high_energy rolled back to geant4-03-02-cand-01.
06/13/2001 11:46	stesting	geant4	geant4-03-02-cand-01	Global	Test1		set to global for system testing
06/06/2001 20:51	jwellisc	geant4/ source/ processes/ hadronic	had-V03-02-00	Selected	Test1		Tag for the 3.2 release.
06/06/2001 20:50	jwellisc	geant4/ source/ processes/ hadronic	had-V03-01-00	Internal	CVS		Collects all previous work. Includes an upgrade of angular distributions and
06/06/2001 11:20	gcosmo	geant4/ source/ materials	materials-V03-01-01	Accepted	Test2	OK	Coworks with "global-V03-01-01".
06/06/2001 11:18	gcosmo	geant4/ source/ global	global-V03-01-01	Accepted	Test2	OK	Added contructor to G4DataVector with additional argument
06/06/2001 10:37	gcosmo	geant4/ source/ tracking	tracking-V03-01-02	Accepted	Test2	OK	Cleared warnings on G4SteppingManager detected
06/05/2001 21:20	pia	geant4/ tests/ test17	test17-V03-01-02	Accepted	Test2	OK	Identical to test17-V03-01-01, except for the new reference outputs compatible
06/05/2001 21:08	pia	geant4/ source/ processes/ electromagnetic/ lowenergy	emlowen-V03-01-17	Accepted	Test2	OK	Bug fix in antiproton ionisation.
06/05/2001 17:49	larazb	geant4/ source/ processes/ hadronic/ models/ generator/ pre_equilibrium	pre_equ-V03-01-00	Internal	CVS		
06/05/2001 17:47	larazb	geant4/ source/ processes/ hadronic/ models/ generator/ de_excitation	de_excitation-V03-01-00	Internal	CVS		
06/05/2001 12:00	johna	geant4/ source/ visualization	vis-V03-01-04	Internal	CVS		
06/01/2001 15:59	stesting	geant4/ tests	tests-V03-01-02	Internal	CVS		

Software Maintenance

- Adoption of standards
- Encapsulation of components
  - minimise coupling to reduce software complexity
  - regular monitoring of architectural dependencies
- Avoid system-dependent solutions in the source code as much as possible
  - centralise system configuration management
  - modular structure for architecture setups
- Avoid “naïve” use of programming language features to maximise porting
- Traceability of updates
  - history files & regular tagging
  - disentangle development from bug-fixes
- Example: *kernel* - see next talk

Customer Support

- Terms of the User Support are defined in the Memorandum of Understanding (MoU)
- Effort shared among WGs
  - contact persons defined for each WG
  - acting as experts in their specific domain
  - joint meetings with users and developers
- Problem Tracking System (*Bugzilla*) available to users
  - flexible design allowing easy customisation for Geant4
  - tokens automatically assigned to responsible persons
  - 260 reports submitted since tool in production
  - ongoing process improvement
- On-line documentation, training and FAQ on Web
- Source code and binaries available on Web and AFS
- *Hypernews* user forum available soon


File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://wwwinfo.cern.ch/asdcgi/geant4/problemreport/query.cgi
Geant 4
Geant4 problem tracking system v0.1 is based on Bugzilla.
Query Page
If you do not select a choice in a category, the default is to report all problems!
Find a problem report that contains these words in the summary, description, file or URL:
Summary: Substring Regexp
Description: Substring Regexp
Submit query
URL: Substring Regexp
File: Substring Regexp
Provide additional information for searching (if you want):
Changed in the last days.
Program: Release: Tag: Component:
Geant4
Geant4 1.0
Geant4 1.1
Geant4 2.0
Geant4 3.0
Geant4 3.1
geant4-01-01
geant4-02-00
other
sometag
analysis
config
digits+hits
digits+hits/detector
digits+hits/digits
Submit query
Status: Resolution: Platform: OpSys: Priority: Severity:
NEW
ASSIGNED
REOPENED
RESOLVED
VERIFIED
CLOSED
FIXED
INVALID
WONTFIX
LATER
REMIND
DUPLICATE
WORKSFORME
All
DEC
HP
PC
SGI
Sun
Other
All
Windows NT
AIX
HP-UX
IRIX
Linux
OSF/1
P1
P2
P3
P4
P5
critical
major
normal
minor
trivial
enhancement
Submit query
Email: matching as substring
Assigned To
Reporter
Submit query
100%
Document: Done.

Documentation

- Six user manuals available on-line
  - Introduction to Geant4
  - Installation Guide
  - User's Guide for Application Developers
  - User's Guide for Toolkit Developers
  - Physics Reference Manual
  - Software Reference Manual
- User examples: novice, extended, advanced
- Training kit: three module-structured courses
- Design documents
- Defined policy for update
  - currently under revision for user docs

Configuration Management - releases

- Defined policy for *major* and *minor* releases
  - 4 major releases, 4 minor releases, 6 patches published since in production (December ‘98)
  - policy periodically revised and updated
- Development releases distributed monthly to collaborators and developers
  - additional development releases if necessary
- Close collaboration with System Testing Team
  - acceptance tests, part also of system tests, are also run independently by the release manager
- Prompt collaboration from developers required during the public release phase

Problem Resolution

- Problem Tracking System tool
  - provides all necessary fields useful for an efficient analysis and tracking of a problem report
  - outstanding problems detected during system testing are posted in the system
- Defined policy for documenting development tags of the source code
  - when submitting code to STT
  - when tags are rejected by STT
  - when generating release-notes of public or development releases
- Frequent tagging of the source code
  - distinguish between tags for grouping consistent new development and tags for inclusion of bug-fixes
  - *CVS* branch tags are adopted if required

Process Assessment

- Define an assessment method
  - adopt a standard model: *ISO/IEC-15504 (SPICE)*
    - identify the scope of the assessment
    - plan the assessment for each individual component
    - validate the retrieved information
    - identify strong and weak areas
    - archive and version the results
    - identify priorities for improvement from final ratings
  - formal assessment based on *SPICE-99* in 1998
  - applied in 2000 to the *software design* process
  - leads to Software Process Improvement
  - based on a valid and complete reference model for software engineering practices

Process Assessment

- Applied in 2000 to the *Software Design* process
  - attempted *level 3* capability. Reached *level 2*
  - one external certified assessor invited, one internal
  - questionnaire created and distributed by e-mail
    - all WG coordinators interviewed
    - results collected and analysed
  - results and ratings presented (Orsay G4 Workshop)
  - recommendations for improvement recorded and planned
- Goal: identify the well established OOP procedure for design development and maintenance in the *production* phase of the software product

Software Process Improvement (SPI)

- Understand, determine and establish applicable procedures to Software development and maintenance of the software
- Make SPI a Software Process *life-cycle driven*.

⇒ Primary life-cycle processes:

  – guarantee that the code quality will not degrade with time: SPI actions associated with a regular QA activity
  – assure that coupling will not increase with the growing complexity of the software

⇒ Improve overall usability and robustness of applications: improve quality, maintainability and reliability of the code.

⇒ Assure continuity and integration of regular system testing within the normal Software development activity.

Software Process Improvement (SPI)

- (Chosen) Domains of applicability in Geant4:
  - Q/A & Optimisation activity
    - applied to the software product in either global and component domain related context
  - Analysis & Design software cycle
    - identify the well established OOP procedure for development and maintenance
  - Testing
    - assure constant improvement and continuity to system testing
- Action for improvement identified
  - plan for SPI established
  - progressive implementation

Future evolutions

- Make SPI part of the software life-cycle
- Consider monitoring progress of the SPI program
  - regular check-points at *Category-Coordinator meetings*
    - regular update of status:
      - **http://cern.ch/geant4/milestones/software_process**
  - include activities addressing SPI in the Collaboration Workshops
- Iterate new assessments in future
  - extend assessment to uncovered (or partially covered) domains (testing, documentation, Software Management)
  - try improving Capability level