Software Process Improvement in Geant4[*]

G. Cosmo CERN-IT/API, Geneva, Switzerland

Abstract

Applying a Software Process Improvement (SPI) program to GEANT4 [1] represents a rather challenging task, both in terms of software development and organisational matters. The complexity of the software involved, the wide areas of application of the software product, the huge amount of code and Category complexity, and the size and distributed nature of the Collaboration itself are all ingredients which involve and correlate together many aspects of SPI.
In addition, different levels of application of an SPI program must be considered, according to the status of each Category Domain component. Although in *production* and available to the public since December 1998, the GEANT4 software product includes some Category Domains which are still under active development. Therefore they require different treatment in terms of improvement of the development cycle, system testing and user support.
This document summarises the current status of the SPI program applied to GEANT4 and initiated last year, as part of the project's milestones for years 2000 and 2001.

Keywords: SPI; Object-Oriented Design; Quality Assurance; Testing

1 Introduction

The GEANT4 project started in November 1994, when it was approved by the DRDC (Detector Research and Development Committee) as CERN R&D project (RD44) [2]. From 1995 to its completion in 1998 the project has reported to the LHC Committee (LHCC/LCB) meeting the required milestones. The first prototype was delivered at the end of 1995, the first *alpha* version was released in spring 1997 and the first *beta* version was announced at mid-1998.
The first *production* version was delivered at the end of 1998. With this release the RD44 project met its goals and was therefore completed. Since 1999 the Production Service, User Support and development of GEANT4 have been managed by the international GEANT4 Collaboration, which is based on a Memorandum of Understanding (MoU) [3] among the participating Laboratories, Experiments and National Institutes.
The GEANT4 software has been developed via a collaboration world-wide of about 100 scientists, coming from over 40 institutes and experiments in Europe, Russia, Japan, Canada and United States. At this time of writing, it is currently at its fourth major production release distributed to public.

2 Software Process Improvement Goals

By Software Processes it is intended the set of processes used by an organization or project to plan, manage, execute, monitor, control and improve its software related activities. Software Processes define the practices that are used in the production and evolution of the software.
The main goal of the SPI program [4] in GEANT4 is to understand, determine and propose applicable procedures for software development and maintenance in the *Production* phase of the software product. The last assessment on the project (based on the SPICE [5] Model) was performed in October 1998, i.e. during the R&D project phase. We profited from experience of members in the GEANT4 Collaboration to help identifying weaknesses in those areas where to apply SPI. At the same time, we stress in considering SPI as a gradual process to apply along with the software life cycle and being indeed "life cycle driven". We expect to continue this

[*] Status Report - January 2001

activity, by performing assessments in future to address those areas previously not investigated and to monitor progress of the SPI program itself.
Three main areas of application of SPI in GEANT4 were chosen in the current phase:
Object-Oriented Analysis & Design (OOAD) software cycle, Quality Assurance & Optimisation (QA) and System Testing.
Concerning OOAD, the goal is to guarantee that the code quality will not degrade with time and assure a coherent development where coupling will not increase with the complexity of the software, by applying the suggested SPI actions associated with a regular QA activity.
Actions suggested for QA embrace several activities to be regularly applied, both in a global context and within each component domain of the project. These activities aim to improve overall usability and robustness of the applications at run-time and to improve quality, maintainability, portability and reliability of the code in general.
Actions suggested for testing aim to improve the System Testing activity itself, assure its continuity and integration with the normal software development. A well organised and well integrated System Testing activity will help indirectly to improve the quality of the user support by frequent releasing of well tested code.
Establishing well defined methods and procedures is of vital importance for GEANT4, whose mandate is to provide a software product and maintain it over a reasonably long life-time with continued good reliability and robustness.

3 Software Processes

Many software processes may be addressed by SPI arising from various process categories: primary life-cycle of software development, supporting life-cycle, management processes, organisational life-cycle and user-supplier processes. Processes belonging to each category may be correlated. Therefore the correlations must be taken into consideration for any assessment.
Among these categories we can identify the following processes (the complete list of software processes in ISO 15504 [5] can be found in appendix A):

- Development or Engineering processes: system & software requirements analysis, software design, software construction, software integration and unit testing, software maintenance
- Documentation
- Configuration and Change Management
- Problem Resolution
- Quality Assurance and Measurement
- System Testing, Acceptance and Releasing
- Verification and Validation
- Reviews, Audits and Joint Reviews
- Project Tasks Management
- Risk Management
- Improvement Process
- Process Establishment
- Human Resource Management
- Infrastructure
- User Support, Distribution

A particular process can be deployed at different levels of generality. Tailoring of processes is required sometimes for different domains because of quality, of stability requirements, or due to the evolution phase of a specific domain, or in order to adapt a process to the people. SPI is a process which must be applied with the full support of all parties concerned, gradually and after identifying the right priorities and objectives [8].

4 Current Status in Geant4

Most of the procedures and methods of application of software processes mentioned in this section derive from the RD44 project [2] specifications. They were applied during the development phase of the project, but are still valid especially for those domains of the project where development is in progress.

4.1 Primary life-cycle processes

The life-cycle model adopted for most domains in GEANT4 is both iterative and incremental (also called *spiral* approach) [9]. The steps among Analysis of Requirements, Design, Implementation and Testing are repeated. Refinements and extensions to detailed design and sometimes also to the architecture have been applied according to new requirements or design issues. In the current Production and Maintenance phase, the life-cycle model is iterative for most domains; it allows the application of successive refinements to the existing architecture, by applying experienced solutions to analysis and design iterations.

- Requirements elicitation process:
  Problem domain and use-case analysis led to elicitation of the User Requirements [10] during the initial phase of the project. User Requirements have been systematically reviewed and updated following the ESA PSS-05 software engineering standard [11]. The User Requirements Document (URD) is now maintained in a source repository providing also automatic versioning; it will be subject to revision during year 2001. Sources of the URD for specific project domains will be also kept and maintained in the repository and available to members of the Collaboration.
- Software Design:
  The Booch (Unified) [9] methodology is employed for Object-Oriented Analysis and Design of the software. The Booch/UML notation has been chosen as the common *language* for documentation of designs and internal design reviews. Old documents in Booch notation exist and are being progressively updated and converted to UML.
  The Rational Rose CASE tool [12] has been extensively used for the initial generation of the design documents and, where required, for reverse engineering. Standard documents provided for architectural and detailed design are:
  - The class *Category Diagram*: showing the overall picture of the problem domain analysis in the project, the use relations between the different class Categories and the dependency flow. A correct domain decomposition and a well considered set of dependencies (avoiding circular use relationships), allows working groups associated to each Category domain to work largely in parallel, establishing also a hierarchy for delivery.
  - *Class Diagrams*: relationships among classes and their main attributes and methods defining the interfaces are shown in these diagrams. Classes are grouped according to Categories or sub-Categories and Class Diagrams represent the structure of the specific sub-system.
  - *Scenario Diagrams* (*Object Diagrams* or *Interaction Diagrams*): show how existing objects relate each other in the logical design of the system. They can be used to illustrate a precise event and show how objects interact through their interfaces in a temporal sequence.
  - *Class Specifications* or *Mission Statements*: describe mission and main (public) functionalities of a generic class. A Software Reference manual is also provided.
- Software Construction:
  Programming and coding guidelines [13] were adopted from the beginning. It was felt important not to impose too fixed rules or style-conventions for a world-wide collaboration

like GEANT4, but just flexible and adequate guidelines basically dealing with adhesion to the object oriented paradigm (data-hiding, encapsulation, etc.), performance, and portability of the software.
Packaging of the software has strictly followed the domain decomposition into Categories and sub-Categories that resulted from the design process. Wherever applicable, classes defining interfaces are packaged in sub-Categories separately from the concrete classes implementing such interfaces. In such a way, classes beloging to a Category collaborate to provide a set of services in a re-usable way.
Tools for Quality Assurance are used, mainly in the areas concerning source code filtering for coding rules violations and run-time memory management. Code filtering is periodically performed (twice a year) in a "global" context, and recently an automatic mechanism for submitting code filtering through WWW using the CodeWizard tool [14], has been introduced. This allows to any developer in GEANT4 to submit unit-package source code filtering and receive the results automatically by e-mail.

- Software Integration and Unit Testing:
  System aggregates that can be tested together are identified according to the dependency structure of Categories. Related tests are regularly monitored as part of the routine testing procedures [17].
  Unit testing is performed indipendently within each Category or sub-Category and tests are implemented trying to maximize coverage as much as possible.
- System Testing, Acceptance and Releasing:
  System Testing activity is deployed by a specialised team, the System Testing Team (STT). Procedures for testing [17] and releasing [18] are defined and strictly applied in order to guarantee regular and efficient delivery of well tested sets of tags including bug-fixes and/or new development. The release procedure foresees that tags for the various Categories are submitted in groups where the order strictly follows the dependency structure defined by the class Category Diagram. Acceptance tests which are also included in routine system tests are built and run separately by the Release Manager during the release phase. Major public releases are distributed twice a year. Bug-fixes are collected and periodically made available as public patches or minor releases.
- Software Maintenance:
  In order to achieve maintainable software and ensure its quality, the adoption of standards, wherever possible, is promoted. Encapsulation of components is maximised, inter-dependency and unit complexity is minimised. We try to assure portability of the software by constantly monitoring the evolution of compilers on different system architectures, and by avoiding adoption of system-dependent solutions or naive language features wherever it is the case.
  Traceability of updates, extensions and bug-fixes to the code is assured by means of maintaining ad-hoc history files, regularly tagging the code and by trying to disentangle routine development from bug-fix updates [18].
- User Support, Distribution:
  The terms of the User Support in GEANT4 are defined in the article 2 of the Memorandum of Understanding (MoU) [3] document. Contact persons for each working group are nominated and are responsible for managing and resolving problem reports submitted by users through the WWW by means of the GEANT4 Problem Tracking System [24], which automatically assigns a problem report according to the specified Category domain affected.
  The GEANT4 WWW site [1] also provides on-line documentation, a FAQ page and the list of contact persons for each Working Group domain. A public User Forum based on Hypernews [25] will also be setup soon.

4.2 Supporting life-cycle processes

- Documentation:
  As user documentation [19], GEANT4 provides six documents (available on-line from WWW) addressing inherently different topics and levels of expertise:
  - *Introduction to* GEANT4: a general introduction to the GEANT4 toolkit, mandate, required computing environment, user support policy.
  - *Installation Guide*: guide to a step-by-step installation of the toolkit and specification of required resources and third party software and tools.
  - *User's Guide for Application Developers*: provides a step-by-step tutorial in the use of GEANT4 for a generic user; it describes the usage of the toolkit for practical applications and also for more advanced uses, providing example codes.
  - *User's Guide for Toolkit Developers*: guide oriented to developers who want to contribute to extend functionalities in the GEANT4 toolkit - like, adding new physics processes, new particle types, etc. It includes a large part of the design documents (mainly class diagrams) related to the implementation of the system kernel. It also includes a guidance on how to extend the functionality of each class category.
  - *Physics Reference Manual*: contains a detailed description of the physics models employed in simulating various kinds of physics processes implemented in the toolkit.
  - *Software Reference Manual*: includes the description of the definitions of most relevant classes and their public interface methods. It's automatically generated from the source code.

  Documentation in the form of comments in the source code is also provided, basically for use of toolkit developers.
  User examples distributed with the toolkit are referenced in the documentation in form of a self-tutorial, starting from "novice" examples showing simple applications of the GEANT4 toolkit, to "extended" examples for specific applications, to "advanced" examples showing real and complete applications and using external extensions for advanced graphics or analysis modules.
  Design documents are maintained in a separate source repository available to developers in the Collaboration, together with the User Requirements documents and the Training Kit Tutorial.
  The Training Kit Tutorial has been developed to address three different kind of courses:
  - *Short 1-day Lecture* - short general course of 1 day;
  - *School-type Course* - course suitable for a school, about 5 hours lectures plus hand-on sessions with exercises;
  - *Academic-style Lectures* - series of lectures focused on specific topics for approximatively 3-5 hours.

  Documents, papers, publications and much more is also available from the GEANT4 WWW site [1].
- Configuration and Change Management:
  - *Software Configuration Management*: A server for software and documentation repositories is in place; it is based on CVS [20] as basic tool for concurrent version management. The code and documents in the repositories are accessible to members of the Collaboration through AFS [21] at CERN and also through "pserver" read/write access. An account at CERN and disk quota on AFS is provided for each project developer. Creation of new packages or repositories and any other technical issue is under control of the Software Management coordinator, who is also responsible for the WWW maintenance.

  - *Tagging and Versioning*: Category coordinators have the responsibility of managing the development within their Categories and provide tags for testing and releases following well specified rules [18]. Coordinators are invited to tag frequently their code, announce the tag to the System Testing Team, with a description of changes included. The STT will then run all validation system tests for all supported compilers/architectures. The Bonsai [26] tool is used as database to automatically detect any new tag introduced into CVS.
    A global reference tag is provided every month, including all tags which passed system validation tests. The tag is announced and made available to developers and collaborating Institutes for development.
    Major public releases are distributed twice a year. Bug-fixes (no new developments) are periodically collected and publicly made available in the form of patches or minor releases.
    User documentation is tagged according to the major public releases.
- Problem Resolution:
  A customised version of Bugzilla [27] is used in GEANT4 as a Problem Tracking System [24] for bug reports and feedback. The system, accessible from WWW, is open to users and is set up to automatically assign problem reports to responsibles according to the Category domain affected. A report can contain all necessary and useful information for a correct and efficient analysis and tracking of the problem posted.
  Developers are encouraged to always document changes committed in the source repository, either with the CVS command itself and by properly recording any activity in appropriated "History" files associated to each Category. Tags announced to the System Testing Team must be properly documented and regular release notes at every minor/major release or global development tag are provided.
  Category Coordinators are also invited to distinguish as much as possible tags including bug-fixes from tags including new development; branch tags in CVS are sometimes required and used.
- Quality Assurance and Measurement:
  Code walk-throughs are periodically performed through specialized tools for monitoring against violations of established coding rules. The CodeWizard tool [14] has been used for this purpose and an automatic mechanism for submitting code filtering for unit Categories has been put in place and is available to GEANT4 developers.
  Checks on run-time memory management are regularly performed before every major public release; tools like Insure++ [15] and SUN Workshop [16] are used over selected test-bed applications.
  Checks for violations of the dependency structure of Categories at macro level are performed periodically and correspondance with the main class Category Diagram is monitored.
  Performance monitoring on selected test-bed applications is applied at unit level for Categories where performance is critical. Results are compared against previous measurements based on stable releases of the code.
- Verification and Validation:
  At macro level, the dependency structure of Categories is verified against the main class Category Diagram taken as reference.
  General functionalities of the Toolkit are verified at every new revision of the URD [10]. Verifications of functionalities and coverage at unit level are under responsibility of each Category Coordinator, as well as unit tests and validation of new developments or fixes.
  New development is validated by the STT once all system integration tests have been successfully performed, provided that, in collaboration with Category Coordinators,

system tests have been extended to cover also the new introduced functionalities.

- Reviews, Audits and Joint Reviews:
  Internal reviews concerning technical matters and proposals from users, developers or collaborators are performed during meetings of the GEANT4 Technical Steering Board (TSB), which is organised 6 times per year. Minutes of the meetings are publicly distributed. Joint reviews with users involved in the GEANT4 joint projects are organised periodically. Reviews internal to each working group associated to a specific Category are organised and under responsibility of each Category Coordinator.
  Global reviews of the product or the Collaboration in general are organised by the Collaboration Board (CB), which usually meets twice a year. Management of resources and monitoring of responsibilities or analysis of proposals from the TSB are all matters handled by the CB, according to the GEANT4 Memorandum of Understanding [3].
  Every year, the GEANT4 Collaboration meets for a workshop of one week, where project milestones are reviewed, problems and/or hot topics are discussed, working sessions are organised.

4.3 Organisational life-cycle processes: Management processes

- Project Tasks Management:
  Management of the project is under control of the GEANT4 Collaboration as specified and dictated in the GEANT4 Memorandum of Understanding [3]. Geant 4 is organised in working groups; each working group is responsible for one specific domain of the toolkit where well specified tasks are associated to it. Each working group is represented in the TSB by its Coordinator(s).
  Objectives are defined every year, discussed within the TSB meeting and presented to the CB. They are reviewed generally during the TSB meetings, joint reviews or in the Collaboration Workshop organised once every year.
- Risk Management:
  Technical risks involving flexibility and usability which may affect mainly the architectural design, or risks related to performance penalties, code duplication, code quality and porting which in large part applies to detailed design, are controlled and mitigated in general by the development process adopted. A well established “Macro” process can mitigate those risks (especially those that may affect the “Micro” processes) and assure the success of the project.
  Non technical risks like those related to code delivery, resource management and human relationships are strictly related to tasks planning, where the established schedules must not be overly optimistic.

4.4 Organisational life-cycle processes: Organisation processes

- Improvement Process:
  Principles and goals of SPI are described in this document. A plan [4] was formally presented at TSB meetings and approved as one of the milestones for year 2000-2001. Priorities and objectives were identified and the approved SPI program is currently being applied; some of the actions suggested have already been taken. As expressed in Section 2 of this document, SPI is a process which must be gradually applied; SPI must be considered life-cycle driven, therefore progresses of the established program will be constantly monitored.
  - Process Establishment:
    General process standards where established since the R&D phase [2] of the project and gradually extended or complemented. Their application may vary according to

the phase of the project and its scale, in which case they are reviewed. Application of these standards needs to be regularly assessed.
Reaching the capability level of a performed process establishment /citespice1 in the project is a key objective for GEANT4, also for what concerns an SPI program applied to it.

- Human Resource Management:
  Teams are composed according to the domain decomposition dictated by the design, forming a working group. People are made part of a team according to their interests, expertise and availability. The addition of new members or creation of new working groups must be approved by the TSB and acknowledged by the CB, after the provision of a formal proposal and plan of work. The CB is also responsible for the overall allocation of resources. Organisation in terms of internal training for new members developers and tasks planning is under control of the coordinator of the working group.
- Infrastructure:
  As specified in the GEANT4 MoU document [3], the necessary resources for travel to meetings as well as for software development tools and infrastructure are provided by the parties or institutes of the collaborators, according to the tasks assigned to the working group.

5 Actions for Software Process Improvement

In this section, we list for each class of processes, those Software Processes in GEANT4 which we identify suitable for an improvement; for these processes appropriated actions are suggested. Among all the actions listed below, priority has been given to those mentioned in the executive summary.
Note that some of the actions listed below are already in use.

5.1 Primary life-cycle processes

- Requirements elicitation process:
  - Periodically review and if necessary update the general User Requirements Document (URD), by analysing it also in the context of each category domain, possibly starting from the list of “use-cases”.
  - Adopt automatic versioning for the general URD and category domain specific documents, by keeping sources in a dedicated repository for documents under CVS [20] and structuring it according to domains.
  - Category coordinators should periodically review and, if necessary, update domain specific URDs, where applicable.
- Software Design:
  - Within each Category domain, Category Coordinators should periodically perform the following actions:
    - Review and identify those areas where an OOAD software cycle needs to be applied and implement it;
    - Review design documents (class diagrams, class specifications for the Software Reference Manual, and scenario diagrams for the most relevant object interactions concerned). If necessary update and integrate them;
    - Review the code to check its consistency with the design.
  - Improve for most Category domains the traceability between requirements, design and test-cases, by generating precise mapping between:
    - unit tests and classes as defined in the class diagrams;
    - integration tests and categories as defined in the architectural diagrams;

        * system/acceptance tests and requirements/use cases.

- Software Integration and Unit Testing:
    - Establish clear responsibilities for maintenance and integration of system ans acceptance tests in the normal development process, in order to improve communication and collaboration between the System Testing Team and developers.
    - In order to easily monitor the evolution of the system tests and verify that the required functionalities are correctly implemented and tested: review and properly document current system tests; check and possibly increase the scope of the system integration tests by verifying correspondance with URD and use cases.
    - Improve unit testing coverage also through adoption of specialized tools for coverage analysis. The STT should generate a map for testing coverage in collaboration with Category Coordinators and establish a valid method for its maintenance.
- System Testing, Acceptance and Releasing:
    - Review currently dedicated resources available for testing: manage recruitment of new manpower and promote a training activity for "new-comers".
    - Improve *regression tests* to detect and understand behavioral changes that may happen by the integration of new development.
    - Improve *statistical tests* to help with verifying the coherence of the results with their logical or physical meaning (statistical distributions) for each test.
    - Improve/implement automation: Bonsai [26], LXR [28] and Tinderbox [29] WWW tools. Promote their proper usage among Collaborators.
- User Support, Distribution:
    - Setup a moderated public users' Forum based on WWW, for discussions on generic GEANT4 topics.

5.2 Supporting life-cycle processes

- Documentation:
    - Collect updated design diagram sources, and define a clear procedure for their maintenance and update. Store design diagram sources (.mdl Rose files) in a CVS repository with restricted access to GEANT4 Category Coordinators. Implement a policy to regularly update them.
    - Make available on web the updated design documents (also update of User's Guide for Toolkit Developers) and define a clear procedure for their maintenance and update.
    - Costantly review and improve code quality of official public examples, in order to facilitate take up and training of users of GEANT4 by providing them with a set of well thought out reference examples.
- Configuration and Change Management:
    - Adopt proper *change management* during both development and maintenance phases when design faults are detected or updates are applied.
- Problem Resolution:
    - Improve automation of current Problem Tracking System and promote its proper usage among Category Coordinators.
- Quality Assurance and Measurement:
    - Create a team of 2 people involved for at least 30% of their time. The team should take the responsibility of:
        * Identifying the proper professional tools to be used associated to topics of: dynamic memory allocation monitoring, perfomance monitoring and

        profiling, source coding rules violations checking, code metrics analysis and test coverage analysis. Consider availability of tools and project resources.
        * Identify the valid set of project-specific rules to be applied for source code filtering and implement the proper filters and scripts to be adopted.
        * Select and/or define the “testbed” applications to be used and maintain/upgrade them along the development, in collaboration with the System Testing Team.
        * Delegate to Category Coordinators the responsibility to monitor, assign to developers and/or implement fixes result of the QA activity.
        * Make publicy available to developers tools, setups and scripts, to allow “unit” QA activity within each project domain.
        * Automate as much as possible (and document) the QA activity, possibly through the WWW.
        * Make available and distribute the results of the filtering and analysis done.
        * Perform a complete analysis regularly every 1 or 2 months, based on the latest suggested Reference Tag.
    - Identify possible resources (tools/people) within external groups which are part of the GEANT4 Collaboration.
    - Improve automation: integrate with tools used for testing.
- Verification and Validation:
    - At macro level, periodically (every 6 months) review the current class Category Diagram and check/verify areas where violations/changes have been introduced.
    - Provide test-coverage for validation of code to be submitted to system testing.

5.3 Organisational life-cycle processes: Management processes

- Project Tasks Management:
    - Within each project’s domain, adopt tools for work-flow management. Plan training for effective usage of the tools.
    - Ensure that the on-going development in each Category is consistent with the design dictated by the design documents by supervising the development activity and, according to available resources, organising proper training for the developers in the team.

5.4 Organisational life-cycle processes: Organisation processes

- Improvement Process:
    - Ensure that SPI is life-cycle driven, by regularly monitoring its progress: periodically iterate process assessments and compare results with previous assessments.
    - Extend assessments to uncovered (or partially covered) domains: testing, documentation, software management.
- Human Resource Management:
    - Within each project’s domain, adopt tools for size/effort estimation. Plan training for effective usage of the tools.

6 Summary

The three main areas of application of SPI in GEANT4 for year 2000-2001 were chosen to be: Object-Oriented Analysis & Design (OOAD) software cycle, Quality Assurance & Optimisation and Testing. For each of these areas an evaluation of the current status has been made, taking into

account correlations on the different aspects/domains of the SPI program, weak points were identified and recommendations for improvement were advanced. For the OOAD software cycle domain, a formal assessment based on ISO/IEC 15504 (SPICE) Model [5], [6] has been performed and results from the assessment [22] up to capability level 2 were used to address items of improvement in the SPI program.
In order to consider SPI as part of the software life cycle and to monitor progress of the SPI program itself, new assessments will be regularly performed in future to address also areas previously not investigated, trying to first improve the current capability level and later-on increase it.
Concerning the OOAD software cycle, a need to improve in most Category domains the traceability between requirements, design and test-cases has been expressed, together with adopting proper *change management* during both development and maintenance phases when design faults are detected or updates are applied. Internal training in form of books, monitoring needs to be improved, in particular concerning the design methodology and the development cycle from User Requirements to Implementation. A repository for design documents is required and methods for providing, publishing and maintaining design documents must be defined and applied. Tools for work-flow management and size/effort estimation should be applied within each domain.
For Quality Assurance and Optimisation (QA), it has been requested to create a specialized team of at least two people sharing 30% of their time. The activity of this new formed QA team will be focused on the “global” context of the project, will have to be possibly independent from the software development itself, will need to be performed in coordination with the System Testing Team (STT) and Release Team, and has to be based on *mutual trust* with developers and Coordinators. It has been also adviced to continue in the effort of improving automation, by possibly integrating with system testing automation. Resources in terms of tools and man-power need to be identified not only within members of the Collaboration but also externally.
Concerning System Testing, it has been stressed the importance of following the formalised procedures expressed in the *release & testing procedures* document [23], according to the established responsibilities for the maintenance and integration of the system tests in the normal development process. Emphasis has been put to apply a review of the existing system tests by checking their correspondance with User Requirements and Use Cases, and by properly document them along development. It has been adviced to the STT to generate a map for testing coverage in collaboration with Category Coordinators and establish a valid method for its maintenance. Progressive improvement in the development of regression and statistical tests has been suggested (where applicable). Concerning System Testing Automation, the adoption of WWW tools like Bonsai [26], LXR [28] and Tinderbox [29] are encouraged and expected to be soon integrated in the normal testing activity.
Committments and deadlines for the implementation of the proposed SPI actions have been established.

APPENDIX A: Processes and Process categories in ISO/IEC 15504

Primary-life cycle

Engineering process category:

- ENG.1: Development process
  - ENG.1.1: System requirements analysis and design
  - 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 and testing
- ENG.2: System and software maintenance

Customer Supplier process category:

- CUS.1: Acquisition process
  - CUS.1.1: Acquisition preparation
  - CUS.1.2: Supplier selection
  - CUS.1.3: Supplier monitoring
  - CUS.1.4: Customer acceptance
- CUS.2: Supply process
- CUS.3: Requirements Elicitation process
- CUS.4: Operation process
  - CUS.4.1: Operational use
  - CUS.4.2: Customer support

Supporting life-cycle

Support process category:

- SUP.1: Documentation process
- SUP.2: Configuration Management process
- SUP.3: Quality Assurance process
- SUP.4: Verification process
- SUP.5: Validation process
- SUP.6: Joint Review process
- SUP.7: Audit process
- SUP.8: Problem Resolution process

Organisational life-cycle

Management process category:

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

Organisation process category:

- ORG.1: Organisational Alignment process
- ORG.2: Improvement process
  - ORG.2.1: Process establishment
  - ORG.2.2: Process assessment
  - ORG.2.3: Process improvement
- ORG.3: Human Resource Management process
- ORG.4: Infrastructure process
- ORG.5: Measurement process
- ORG.6: Reuse process

References

1 http://cern.ch/geant4.

2 S.Giani et al., *GEANT 4 : An Object-Oriented Toolkit for Simulation in HEP*, CERN/LHCC/98-44, November 1998.

3 http://cern.ch/geant4/organisation/MOU.html.

4 http://cern.ch/geant4/milestones/software_process/OOAD-items.html.

5 ISO/IEC Joint Technical Committee 1 (JTC1), *ISO/IEC DTR 15504 Software Process Assessment*, Project Editor: Terry Rout.

6 ISO/IEC Joint Technical Committee 1 (JTC1), *ISO/IEC DTR 15504-5 Part 5: An Assessment Model and Indicator Guidance*, Product Editor: Jean Martin Simon.

7 M.Paulk et al., *The Capability Maturity Model: Guidelines for Improving the Software Process*, Addison-Wesley Publishing Co., 1995, ISBN 0-201-54664-7.

8 D.A.Reo et al., *Measuring Software Process Improvement: there's more to it than just measuring processes*, ESI - FESMA 99, September 1999.

9 G.Booch, *Object-Oriented Analysis and Design with Applications*, The Benjamin/Cummings Publishing Co., 1994, ISBN 0-805-35340-2.

10 RD44, *GEANT 4 User Requirements Document*, CERN, 1998.

11 ESA, *Guide to User Requirements Definition Phase*, ESA PSS-05, 1994.

12 I.White and M.Goldberg, *Using the Booch Method, a Rational Approach*, Rational Rose Corp., The Benjamin/Cummings Publishing Co., 1994, ISBN 0-805-30614-5.

13 http://cern.ch/geant4/collaboration/coding_guidelines.html.

14 http://www.parasoft.com/products/wizard/index.htm.

15 http://www.parasoft.com/products/insure/index.htm.

16 http://www.sun.com/forte/cplusplus/.

17 http://cern.ch/geant4/working_groups/testing/testing.html.

18 http://cern.ch/geant4/working_groups/testing/tag_release_policy.html.

19 http://cern.ch/geant4/G4UsersDocuments/Overview/html/index.html.

20 P.Cederqvist et al., *Version Management with CVS*, Signum Support AB, 1992.

21 *The AFS User Guide*, IBM Transarc Corp., April 2000.

22 G.Cosmo, G.Pawlitzek and H.P.Wellisch, GEANT4 *Design Process Assessment*, Internal Note, September 2000.

23 http://cern.ch/geant4/working_groups/testing/testing.html.

24 http://wwwinfo.cern.ch/asd/cgi-bin/geant4/problemreport/.

25 http://www.hypernews.org/.

26 http://bonsai.mozilla.org/.

27 http://bugzilla.mozilla.org/.

28 http://lxr.linux.no/.

29 http://tinderbox.mozilla.org/.