Software Process Improvement
overview

Gabriele Cosmo

CERN IT/API-SI

Gabriele.Cosmo@cern.ch

Outline

- The planned objectives
- The current status
- Planned activity for this Workshop
  - focus on User Requirements
  - User Requirements & Traceability
  - future evolutions
- Conclusions

The chosen domains

- Q/A & Optimization 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

Q/A & Optimization - actions

- Creation of Q/A specialized team (2 people, ≥30%)
  - "global context" activity deployed to the team
  - activity possibly independent from development
  - activity in coordination with the System Testing & Release Teams
  - Team's responsibilities:
    - identify Q/A tools, also considering availability & resources
    - select *test-bed* applications and care for maintenance/upgrade in collaboration with STT and Category Coordinators
    - perform a complete analysis every 1-2 months and delegate to Category Coordinators fixes to the code
    - identify coding rules and implement them through scripts/tool
- Identify resources (tools/people) from external groups in the Collaboration
- Improve automation: integrate with tools for testing

Q/A & Optimization - activities status

- Adoption of specialized tools and scripts
  - → Monitoring of dynamic memory allocation
    (*improve overall usability and robustness of the applications at run-time*)
    - ⊗ test-bed applications selected from system tests
    - ⊗ release coordinator performs filtering in the global context
  - ? Performance monitoring and profiling
    (*improve run-time performance and avoid redundancy*)
    - ⊗ identify test-bed applications, define responsibilities
  - → Source code filtering for conventions and coding rules violations
    (*improve quality and reliability of the code*)
    - ⊗ implement project's specific coding rules to apply to scripts/tool
  - ? Source code filtering for metrics analysis
    (*improve quality, maintainability and portability of the code*)
    - ⊗ define responsibilities and methods, identify tools
  - ? Test coverage analysis
    (*improve quality of testing*)
    - ⊗ identify test-bed applications
    - ⊗ identify tools and methods (global context: associate to Testing ?)

Analysis & Design cycle

Goal: *guarantee that the code quality will not degrade with time. Assure a coherent development where coupling will not increase with the complexity of the software*

- General Actions:
  - ? Review of the global category diagram & URD
    - regular check for violations/changes and additions
    - regular maintenance of general URD
  - → Actions to be performed by Category Coordinators
    - periodically review URD, possibly starting from "use cases"
    - review/identify areas where A&D software cycle need to be applied
    - review consistency of code with design
    - supervise Category activity and organize training
  - → Collect architectural/detailed design and URD documents and define a clear procedure for maintenance and update
    - → Currently in place: CVS tree *documents* on AFS
    - → Notify SW-Management Coordinator for new diagrams/docs on Web

OOAD - actions

- Improve traceability or requirements with design and test-cases
- Promote internal training (books, monitoring, ...)
  - requirements ↔ design ↔ implementation cycle
  - design methodology and CASE tools
- Provide/adopt tools for work-flow management, size & effort estimation
  - training for effective usage of the tools
- Define standards for design documents to be provided and published AND maintain them !
  - CVS repository for design documents (architectural/detailed sources, URD, specifications, ...)
- Adopt *change management* for design faults/updates during development/maintenance

Status of the documents repository

System Testing

- Improvement of system and validation tests
  - → establish clear responsibility for maintenance and integration of tests in the normal development process
    - → formalized in release&testing procedures document
    - → keep updated list with descriptions
  - ? review and properly document tests; check correspondence with URD and use-cases
    - identify most common use-cases
    - create map for testing coverage and keep it updated
    - involve category coordinators
  - → adopt/improve regression and statistical tests
    - establish methods and tools
    - provide a clear time-table for deliverables

System Testing - automation

- adoption of *Bonsai* to automate testing activity and CVS tags submission through Web
  - system already in production (see our *tag-database*)
- adoption of *LXR* for online browsing of code through Web
  - first prototype implemented and available (KEK, TRIUMF)
- adoption of *Tinderbox* to allow developers and testers to monitor progress of system tests and allow distributed control
  - first working prototype already available from TRIUMF
- integrate Q/A automation to provide developers a way to perform basic Q/A checks on code before submitting to test
  - implemented scripts for automating CW code filtering

  (*Negative Note: rarely used by developers since in production*)

Activity on User Requirements

- Goals
  - Establish a way to collect and manage user requirements, in terms of:
    - capturing of new requirements
      - formalize requirements proposal from users community
      - formalize requirements analysis by WG coordinators
      - tracking their implementation & testing
    - collecting and archiving existing detailed requirements
      - starting from the general URD (WG coordinators)
  - Automate implementation of *traceability* of requirements to design and system tests

The form template

- Split in two sections
  - *user's section* : filled by the user's representative or the user advancing the requirement
  - *analyst's section* : restricted to the analyst, responsible to evaluate and then propose to the TSB the new requirement for approval
- Implemented as a Web form
  - standalone *perl/CGI-bin* script
  - supported by a tool (e.g. *Bugzilla*)

Template: the user's section - 1

- Name and E-mail:
  - *Who raised the requirement.*
- Title:
  - *A short title for the requirement, like the subject matter in an e-mail.*
- Description:
  - *A textual description for the requirement.*
- Rationale:
  - *A justification for the requirement.*
- Supporting use-case:
  - *describe one or more use cases that require this.*
    - <use-case Name - brief description>
    - <use-case Name - brief description>
    - <use-case Name - brief description>

Template: the user's section - 2

- Responsible category:
  - *Name of the category this requirement is assigned to.*
- Fulfillment criterion:
  - *A prescription of how to test that the requirement is fulfilled.*
- Relevance:
  - *Degree of happiness if this is successfully implemented <who cares, happy, extremely pleased>.*
- Irrelevance:
  - *Degree of unhappiness if this is not implemented <hardly matters, inconvenienced, extremely displeased>.*
- References:
  - *References to material that support this requirement, or provide supplementary information.*

Template: the analyst's section - 1

- Status:
  - *What is the status of this requirement? High level description, detailed description, design complete, coded, tested, regression test in place.*
- Target completion date:
  - *Priority. When you think this will be released.*
- Related requirements:
  - *List of other requirements that depend on this one.*
  - *List of other requirements, this one depends on.*
- Verifiability:
  - *Has its feasibility been verified - in terms of possible incorporation in the design, implementation in the software, testing of its implementation ?*
- Conflicts:
  - *Other requirements that cannot be implemented, if this one is considered. Include the reason and constraints.*

Template: the analyst's section - 2

- View of classes implementing this requirement:
  - *A collaboration or class diagram showing all the classes whose objects interact to implement this requirement (including the interfaces).*
- Scenarios (optional):
  - *List the scenarios of this requirement.*
  - *Include a brief description of each one.*
    - <Scenario Name - brief description>
    - <Scenario Name - brief description>
    - <Scenario Name - brief description>

The exercise at this Workshop

- for WG coordinators and users' representatives
  - try expressing recent user requirements (for example those that have been presented recently at the TSB meetings) according to the template
- for each WG coordinator
  - try preparing a list of existing requirements implemented now or in the past
    - requirements not expressed or documented
    - from the requirements in the general URD derive those relevant to the WG, and create from them some detailed requirements using the template

Conclusions

- Agreed SPI actions "slowly" progressing:
  - Q/A and Optimization, Analysis & Design cycle, Testing
- Monitor progress of SPI program:
  - through regular Category-Coordinators meetings
  - iterate new assessments in future
    - extend assessment to uncovered (or partially covered) domains (testing, documentation, Software Management)
  - try improving Capability levels
- Assign manpower for future SPI activities
  - member in the Collaboration/Group
  - external consultancy