Requirements, Configuration and Deployment Management Software Selection Considerations

December 15, 2004

Jane Curry
Ben Van De Walle
Background

The evolution and continuous morphing of HL7 tools to satisfy an increasing number of tooling requirements has resulted in a more systemic, global and virtual approach towards HL7 tools development. A growing user community, maturing methodologies, new standards, new technologies, the need to exchange information between tools and the availability of additional development resources are driving these requirements.

Beginning with one HL7 tool to a collection of tools, a shift is occurring towards the development of an HL7 Tooling Architecture with supporting principles, design guidelines, and requirements, configuration and deployment management software. This shift has created a need to fill the gaps and to meet rapidly changing needs for (a) management software, (b) the use (evaluation and selection) of available commercial off-the-shelf software (COTS) and (c) the ongoing custom development of tools.

Some of the latest and new standards from Standard Development Organizations (SDO) such as W3C, OASIS, ISO, OMG, IEEE, ANSI and others are being leveraged and/or incorporated in the development of HL7’s standards and specifications. The vendor development tools supporting these new standards take some time to be made commercially available for HL7 and others, creating a situation where some of HL7’s tooling needs are currently being met by customized solutions.

From moving data, to messages, to semantic information over the wire, the standards, technologies and vendor tools to support these maturing methodologies, such as HL7’s, are still undergoing significant and rapid transitions to include web services, Service-Oriented Architecture (SOA), Model-Driven Architecture, UML 2.0 and others.

HL7 tools today cover a wide spectrum, including:

- HL7 Version 2,
- HL7 Version 3,
- the HL7 Development Framework (HDF) for specification development,
- the visual design of models and specifications,
- the Model Interchange Format (MIF) for enabling the exchange of information between models and tools,
- the publishing of the ballot specifications for feedback and approval,
- the storage of HL7 artifacts in a registry,
- electronic services
- and others.

These assist the HL7 Inc. organization with its 30+ Technical Committees and Special Interest Groups and the HL7 International Affiliates in supporting the rapid collaborative development of health-related global standards.

This document was commissioned by the U.S. Department of Veteran’s Affairs to assist HL7 in developing some guidelines to procuring (product evaluation and selection) and/or developing the tools to support a dynamic, productive and an increasing inter-dependent standards and specifications environment within the HL7 Development Framework and HL7 Tooling Architecture Management and Policy Framework (see
References below). The specific objective is for to select software for the management of Requirements, Configuration and Deployment of HL7 recognized tooling.
Requirements

“The HL7 Tooling Committee is responsible to the HL7 Board for the openness and effectiveness of the processes used to identify, select, develop and support tools and for the resources implications of selecting and using those tools.”

HL7 wants to ensure the continued alignment of and to maximize the opportunity of leveraging (a) interim work products and (b) HL7 managed/sponsored/developed tools. To that end, the following are considerations:

- Development and maintenance of internal tooling to support the standards development process.
- Activities to support continued maintenance and support of the HL7 tooling.
- Tooling requirements from multiple perspectives.
- Challenge to understand the current set of requirements.
- Difficulty to contribute to or comment on the importance of identified requirements.
- Enhance the process of collecting requirements.
- Continue to evolve the process to prioritize requirements.
- Mechanism to manage and maintain the tooling requirements.
- Enable the contribution from HL7 members toward the development of tools to support those requirements.
- Ensure the changes to tooling best address the needs of the general standards-development membership in whole and in part.

Software must support the timely and autonomous decision-making of HL7 Technical Committees and Special Interest Groups in their collaborative and cooperative approach in finding solutions to managing complexity and providing interoperability across multiple health-related organizations.

This document highlights some of the guidelines and/or considerations to assist in the re-use, buy or build decision-making processes for the evaluation and selection of software to support the management of HL7 tooling requirements, configuration, and deployment.
**Intended audience**

The intended audience for this document is primarily those involved in tools development. This document can also be beneficial for the end-user of tools and other stakeholders in assisting to understand the management, control and administration of requirements identification for new or additional HL7 tooling features and functions to be developed and deployed.

**High-level perspective**

Terms are in *italics* are defined in the Glossary section in Appendix A

A high-level perspective of Requirements, Configuration and Deployment includes:

**Requirements (primary role/responsibility – Project Manager)**
- Documented customer *requirement* with example and perceived solution
- Categorization, triage and prioritization of *requirements*
- Identify category: V2, V3, publishing, MIF, visual designing, etc.
- High-level feasibility analysis to better understand a *requirement*
  - *Project charter* (with *work product* deliverables or change request(s))

**Configuration (primary role/responsibility – Prime Committer/Developer)**
- *Configuration item or work product* requirements
- Detailed analysis and design
- Re-use, buy or build decision
- *Configuration item or work product* testing – test cases, scripts and data
- *Configuration item* documentation – internal and maintenance

**Deployment (primary role/responsibility – Release Manager)**
- Scheduling
- Quality assurance (validation and *requirements traceability*)
- Integration testing of *package*
- Integration documentation (includes FAQ – frequently asked questions)
- Training (material preparation in scope, delivery out of scope)
- Installation (selection of most appropriate approach for each tool)
- Support (contact list in scope, help line out of scope)
- Education (out of scope)

Project planning and integrated project management applies to all of the above.
The following Figure provides additional context:

**Requirements Management**

1. **A – TC solicits requirements**
   - CVS node created
   
2. **A – Requestor submits change request**
   - Form to be filled out online
   - “Date required by” field
   - “Showstopper” field

3. **A – Triage change request (by committer)**
   - Filter bug “show stoppers” – bugs go to committer of code branch involved
   - Committer determines if the

---

**Legend**

- User Community
- Requestor
- Code ok?
- Yes
- Commit changes to code base
- No — return to developer
- Internal or Externally generated?
- Internal or Externally generated
- Acquire external solution
- Investigate solution alternatives
- Committer determines release targets
- Yes
- No — return to developer
A reported bug is caused by coding error, inadequate documentation or some other source. Triage coordination could be undertaken by the set of committers working as a group. The process will need to evolve as multiple code branches are undertaken. Triage is primarily to determine order of magnitude of requested functionality if associated with existing code base. All other requests go directly to Tooling Committee for prioritization.

- Software requirement to be able to update request with initial estimates of effort and proposed triage resolution.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>D – Urgent action required?</td>
<td></td>
<td>• Showstoppers are addressed on an urgent basis. Changes undertaken to address become subproject with target dates for release.</td>
</tr>
<tr>
<td>4.</td>
<td>A – Committer prioritizes change request</td>
<td>• Committer’s prioritization primarily to estimate degree of difficulty and recommends which code branch to modify. Committers to identify and document criteria.</td>
</tr>
<tr>
<td>5.</td>
<td>A – Membership advises on priorities</td>
<td>• Membership recommends priority setting criteria – probably multi-dimensional involving target audience, nature of expected benefits, alignment with business plan objectives. TC holds open sessions during WG meeting to decide and document criteria – those not present can submit suggestions in writing beforehand</td>
</tr>
<tr>
<td>6.</td>
<td>A – TC prioritizes change requests</td>
<td>• Prioritization criteria needs to be documented and readily</td>
</tr>
</tbody>
</table>


<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>
| 7. | A – TC publishes requirements | • Requirements management software needs to be able to record information about priorities, including history of prioritization changes  
• Acceptance criteria needs to be documented in both general terms and for the specific requirement |
| 8. | A – TC investigates solution alternatives | • TC solicits volunteers to evaluate prioritized requirements and offer suggested solutions.  
• Evaluation criteria for |
|   | D – Internal or external solution (COTS)? | • Either solution requires a project charter with a project leader to follow-through.  
• Requirements updated with project identifier to cross-reference with Configuration Management |
| 9. | A – TC acquires external solution | • Customization of COTS becomes an internal development project |
| 10. | A – Tooling Committee advises on release targets | • Tooling Committee needs to have a current status of all work in progress when recommending release dates and package scope. |
| 11. | A – Committer determines release targets | • Release targets are for packages, which may be comprised of a number of software components, either internally developed. Packages include code, help documentation and training material (including examples)  
• Bug fix releases are modifications to an existing package, but still need to be tested via integration testing to avoid unintended ripple effects |

**Configuration Management**
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>12.</td>
<td>A – Developer requests CVS access</td>
<td>• Online form available via members only website.</td>
</tr>
<tr>
<td></td>
<td>D – HL7 member?</td>
<td>• Membership or authenticated by member organization via providing access to HL7 members only site.</td>
</tr>
</tbody>
</table>
| 13. | A – CVS access granted | • User id initially granted on read-only basis – persons with a CVS account considered a developer.  
• Software provides list of people granted CVS access |
| 14. | A – Developer request new CVS node | • New code branch request may be associated with one or more requirements from outstanding requirements list. Project definition required. Software support needed to manage lists of projects with link to project charter. |
|   | D – Project definition approved by TC? | • Project Definition is probably more like a scope statement, with the full details yet to be worked out. A project charter would evolve as details about the requirements are documented and test cases developed. Expectations are that CVS will manage the full slate of work products associated with a project. |
| 15. | A – TC publishes project definitions | • Project Definitions need to be online and accessible to the general public. The usual **Who** is undertaking **What** work **When** with **What** Benefits to **Whom** and a current status time-stamped. This represents the work currently being undertaken by HL7 and should help assure people with requirements where their needs are in the development cycle.  
• Any project specific acceptance criteria must also |
be documented and available. As projects progress, additional information including test cases, user help documentation, design documentation must also be available online, with the ability to manage multiple versions to remain in sync with the accompanying code.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>
| 16. A – CVS node created | • A node is created either as a sub-branch for an existing code base to accommodate planned enhancements, or a new node for a new code base.  
• The software needs to relate one or more projects to each CVS node as well as identify the set of users who are authorized committers. |   |
| 17. A – Developer submits code to be committed | • Developers can work offline on projects for one or more code bases. Those developers who are not themselves committers must submit their proposed code updates to an authorized committer. The acceptance criteria for code to be committed may need to be negotiated on a case by case basis, but will follow minimum requirements established by the TC.  
• CVS environments manage the versioning of code submitted. |   |
| D – Code OK? | • The set of committers must agree on the validity of the code to be added to the code base. The committers for each code base must manage the coordination of changes among themselves. Ideally, more than one committer is |   |
familiar with the full code base and can provide continuity when any one committer’s availability is inadequate to meet timeliness expectations.

<table>
<thead>
<tr>
<th>18.</th>
<th>A – Commit changes to code base</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• A new version of the code base, with the full set of accompanying documentation is added to the CVS.</td>
</tr>
<tr>
<td></td>
<td>• The committers of a particular code base are responsible for recruiting their “beta testers” and providing access to “pre-release” versions of the code base.</td>
</tr>
<tr>
<td></td>
<td>• The software should assist the committers with tracking beta test users and their feedback.</td>
</tr>
<tr>
<td></td>
<td>• Defect tracking is a desirable feature.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D – Release complete?</th>
</tr>
</thead>
<tbody>
<tr>
<td>• More than one code base may be included in a package. The set of committers contributing their code bases to a package serves as the initial quality assurance team.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Deployment/Release Management</th>
</tr>
</thead>
<tbody>
<tr>
<td>19.</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>D – Release criteria met?</td>
</tr>
<tr>
<td>20. A – Distribute release</td>
</tr>
</tbody>
</table>
| 21. A – Ongoing support | • Ongoing support will evolve as the rest of the process evolves. Any means to assist users to be self sufficient will be desirable.  
• Software that helps manage problem logs, FAQs, self-directed training, in-context help, etc. will contribute to the sustainability of the process.  
• In the near-term, questions directed to the Tooling Committee’s listserve can serve as an informal feedback loop. |

**Assumptions/Goals**

There exist a number of global HL7 user audiences and communities, each with their ongoing needs, challenges and deliverables. These include specification content creators, model and vocabulary facilitators, specification designers, publishers, balloters, early adopters, tools developers and specification implementers. International Affiliates have all the same kinds of audiences but may have additional requirements to localize specifications for their own environments. One complicating factor is that the same people may participate as more than one of the above type of audience or in more than one community.

A few assumptions/goals are included below:
• The process of developing, acquiring and enhancing tools needs to be open, transparent, effective, efficient, relevant, and useful to multiple audiences
• Existence of a source control mechanism to manage multiple specificity levels
• Evolving HL7 methodology and design considerations reflected/enforced by tools
• Tool features/functions to be made available in a timely manner
• Increase self-discoverable understanding and submission of tooling requirements
• Assist in involving more developers in the tooling effort
• If a feature/function does not exist, first research open-source and current COTs markets before committing to HL7 development
• Environments are controlled locally and managed across locations
• The existence of an updated Intellectual Property policy and registry for artifacts
• Platform independence wherever possible for deployed solutions
• No specified IDE (integrated development environment)
• Needs-based configurable tooling and “packages” created from multiple sources
• Tooling enables vendor software interfacing and registration of profiles
• Tooling leverages global standards from Standards Development Organizations
• Web-based for global usability and non-proprietary tool set
• UML 1.4 or higher compliant with ability to extend (current profile – UML 1.4)
  Challenge of different implementations of a specification concept
• Minimize overlap of functionality across configuration items and branches
• Transparency of requirements gathering, prioritizing, development and delivery
  Available multiple views from start to finish
• Requirements related information - MUST
  Captured online, easily collectible (templates), readily exchangeable
  Incrementally added to, changes identifiable, highlight dependencies
• Beta tester test prototypes and software prior to final release of package
• Early adopters are individuals/organizations providing/validating requirements.
• Implementers refer to (a) developers/specification, (b) user/released “packages”
• Consistency and congruency with other HL7 documentation and practices
• Use of a common set of terms defined a glossary

**Out-of-scope**

Tooling covers a broad range of areas and the following exclusions have been identified as out-of-scope relative to the stated “Required” above:
• Research and subsequent recommendations on specific software to use
• Governance-related considerations
• A tool interchange format or the actual development methodology (or processes)
• Training delivery, education and help line

**Constraints**

Constraints include:
• Limited financial funds available for acquiring and maintaining software
• Limited developer resources available
• Limited volunteers available
• Limited time to satisfy an increasing number of requirements
• Communication and semantic complexities
• Absence of selectable glossary by audience or transitional glossary
• Overloaded HL7 terms – e.g. implementation
• Unavailable definitions for some HL7 terms – e.g. project

Desired general characteristics of an integrated software solution

To assist in the evaluation of software for the management of requirements, configuration and deployment software, the starting point is the ideal scenario where one piece of software could satisfy all of HL7’s needs. The reality is that more than one software tool may be necessary to achieve the multiple goals.

The following highlight some of the characteristics and/or features of an integrated software solution:

• Can be installed in a source-controlled and reliable environment
• Secured and managed access – authentication and authorization
• IDE independent
• Ability to identify interdependencies
• Ability to exchange relevant information
• Flexibility, adaptability and ability to customize including filters
• MIF (integration of)
• Selectable features and functions
• Ability to add incremental features and functions
• Ability to use naming conventions
• Ease of use by audience – e.g. GUI and a command line
• Ability to select contents of a package to be released for consumption
• Ability to select deployment based on target environment
• Ability to create and customize documentation, help files and training materials
• Ability to filter (resources/skill sets and capacity/capability)
• Ability to track changes
• Ability to provide/restrict access based on roles and/or declared area of interest
• Ability to capture requirements and their associated project plans and deliverables
• Ability to create “packages” across multiple locations
• Ability to capture roles, responsibilities and accountabilities
• Ability to create workflows

Desired Requirements Management software characteristics

For Requirements Management the following specific software characteristics have been identified as some of the considerations when evaluating software:
• Receive request via the Internet
• Receive with the request any or all of:
  An example of current situation and desired end state
  A statement/criteria for describing “measurement of success”
  A test case and test data
  Rationale for making a change
• Receive request from multiple audiences/users/declared area of interest
  Addition – new feature, function or code base
  Correction – bug or patch
  Modification
  Enhancement
  Re-write/re-factor/re-engineer
  Recommendation to use an off-the-shelf component/software
• Confirm receipt of change request
• Provide additional information to initial change request or resulting project
• Identify the urgency – e.g. show-stopping bug fix
• Capture the
  Risk assessment
  Impact assessment – positive (e.g. productivity) and undesirable
  Availability assessment (re-usable component(s), open source, buy, build)
  Feasibility analysis (time, skills, availability, etc.)
  Total effort by change request to include anticipated future maintenance
• Provide flexible “rack and stack”
• Trace to HL7 overall priorities
• Triage
• Provide choices for:
  Categorization
  Priorities
  Target dates
• Provide various views/reports of selectable global need-specific information
• Identify the acceptance criteria – by change request, by project, etc.
  Standards and Technologies
  Principles
  Backward and forward compatibility
  Documentation
  Training
  Dates
  Costs - total effort, both direct and indirect
• Assign the change request or project

**Desired Configuration Management software characteristics**

For Configuration Management the following specific software characteristics have been identified as some of the considerations when evaluating software:
• Provide read and/or write on a branch/node basis
• Align a configuration item with the Tooling Architecture
• Enforce coding standards
• Identify code base and the prime and other committers
• Identify developers – by artifact, by projects, by change request, etc.
• Identify artifacts – by id, by test cases, by test data, by configuration item
• Identify package – by id, by documentation, by training material
• Identify performance testing – by artifact, by package, etc.
• Identify configuration item by code base/branch
• Track – configuration item, artifact, package, project, documentation, etc.
• Identify configuration item/code/artifact dependencies and interdependencies
• Track verification of base criteria for development
  Coding standards
  Documentation standards (internal and external)
  Unit test standards
  Architectural alignment
  Alignment with prioritized requirements
• Track verification of package with end-user acceptance criteria
• Track patches, bug fix, minor and major releases, packages, etc.

**Desired Deployment Management software characteristics**

For Deployment Management the following specific software characteristics have been identified as some of the considerations when evaluating software:

• Provide for the tracking of adequate Quality Assurance
• Delivery of a package to distributed end-user location(s)
• Create a readily installable package
• Notify user when package is ready
• Indicate the availability of a created package
• Access created package(s)
Software Considered and Selected

The following tables reflect the candidate software that was reviewed by Tooling Committee volunteers. Most met a significant subset of the selection criteria. Rather than go through an extensive criteria documentation and rating exercise, volunteers investigated the short-listed and presented their recommendations based on the critical criteria of ease of use, web based, integrated development environment and low cost.

### Table 1: Configuration Management

<table>
<thead>
<tr>
<th>Product</th>
<th>Source Control (SCM)</th>
<th>Defect and Change Tracking</th>
<th>Client</th>
<th>Integrated (SCM+DCT)</th>
<th>Open-Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVS</td>
<td>✓</td>
<td></td>
<td>WinCVS, Eclipse</td>
<td>Eclipse, Bugzilla</td>
<td>✓</td>
</tr>
<tr>
<td><a href="http://www.cvs.org">www.cvs.org</a></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bugzilla</td>
<td></td>
<td>✓</td>
<td>Web-based</td>
<td>CVS</td>
<td>✓</td>
</tr>
<tr>
<td><a href="http://www.bugzilla.org">www.bugzilla.org</a></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stellation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td><a href="http://www.eclipse.org/stellation/">www.eclipse.org/stellation/</a></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GFForge Project</td>
<td></td>
<td>✓</td>
<td>Web-based</td>
<td>✓ Provides additional open-source project management using free tools</td>
<td>✓</td>
</tr>
<tr>
<td><a href="http://www.gforge.org">www.gforge.org</a></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Perforce</td>
<td>✓</td>
<td>✓</td>
<td>Proprietary</td>
<td>Eclipse, Bugzilla</td>
<td>✓</td>
</tr>
<tr>
<td><a href="http://www.perforce.com">www.perforce.com</a></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rational ClearQuest/ClearCase</td>
<td>✓</td>
<td>✓</td>
<td>Proprietary, Eclipse</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Borland StarTeam</td>
<td>✓</td>
<td>✓</td>
<td>Proprietary</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><a href="http://www.borland.com/starteam/">www.borland.com/starteam/</a></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SourceForge Enterprise Edition</td>
<td>✓</td>
<td>✓</td>
<td>Web-based</td>
<td>✓ Provides additional project management</td>
<td>Contact vendor for quote</td>
</tr>
</tbody>
</table>
Table 2: Development and Build

<table>
<thead>
<tr>
<th>Tool</th>
<th>IDE</th>
<th>Scripting, Building</th>
<th>Integration</th>
<th>Open-Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eclipse</td>
<td>✓</td>
<td>-- integration with</td>
<td>CVS, ClearCase, Perforce, Ant – provides build and scripting</td>
<td>✓</td>
</tr>
<tr>
<td><a href="http://www.eclipse.org">www.eclipse.org</a></td>
<td></td>
<td>Ant</td>
<td>Eclipse, WebSphere, Jbuilder, etc.</td>
<td>✓</td>
</tr>
<tr>
<td>Ant</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 3: Project Hosting Options

<table>
<thead>
<tr>
<th>Project Hosting Option</th>
<th>System Management</th>
<th>Funding</th>
<th>Scripting, Building</th>
<th>Integration</th>
<th>Open-Source Only</th>
</tr>
</thead>
<tbody>
<tr>
<td>SourceForge.Net</td>
<td>Designated volunteers, remote hosting</td>
<td>Voluntary Donations</td>
<td>Ant</td>
<td>CVS, Bugzilla</td>
<td>✓</td>
</tr>
<tr>
<td><a href="http://www.sourceforge.net">www.sourceforge.net</a></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NIAT</td>
<td>Existing MOU with HL7, designated NIAT resources, on-site hosting</td>
<td>Grants</td>
<td>Ant</td>
<td>CVS, Bugzilla</td>
<td>Accomodates the Intellectual Property requirements of HL7</td>
</tr>
<tr>
<td><a href="http://www.nsceee.edu/niat/">www.nsceee.edu/niat/</a></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Software Selection

At the Atlanta Working Group Meeting in September, 2004, the Tooling Committee reviewed the options and agreed to an approach and tentative software selection. The optimal outcome will be a web accessible, open source environment that can support project management as well as Requirements, Configuration and Deployment management support. NIAT is currently the HL7 designated host and the selected software will be downloaded, installed, configured and tested before the selection will be considered final.

The software configuration selected is Gforge Project as the User Interface to the Configuration Management tool CVS with Bugzilla providing defect and enhancement tracking. The Integrated Development Environment is Eclipse integrated with Ant. There are numerous options that must be fine tuned before the environment is ready to deploy and the processes described in the Requirements, Configuration and Deployment Management Process document.

Next Steps and Timelines

Members of the Tooling Committee has explored some of the tools since the Atlanta WG meeting but not completed the necessary installation and configuration. As of December 9, 2004, the following steps and schedule were identified.
1. Acquire current versions of Gforge Project, CVS, Bugzilla, Eclipse and Ant and install at NIAT – December 16, 2004
2. Configure options for initial use – December 16, 2004 - January 6, 2005
3. Pilot by migrating current requirements tracking to Gforge and documenting high priority tooling projects – January 6 - 20, 2005
4. Present to general audience at January WG Meeting in Orlando – January 25, 2005
Appendix A - Glossary

A number of terms are used in this document for which a definition/criteria has been included to assist the reader with the semantics and a context for key concepts and overloaded words. The source of the definitions identified by CMMI is the Software Engineering Institute’s (SEI’s) Capability Maturity Model Integration glossary.

Beta – see prototype
Branch – main line of code; environment dependent
Bug fixes – applies to code; can include various documentation
Change request – MUST be completed prior to ALL changes to code; is a subset of a project charter; can include bug fixes and patches
Configuration item – an aggregation of work products that is designated for configuration management and treated as a single entity in the configuration management process (CMMI).
Configuration control – an element of configuration management consisting of evaluation, coordination, approval or disapproval, and implementation of changes to configuration items after formal establishment of their configuration identification (CMMI).
Configuration management – a discipline applying technical and administrative direction and surveillance to (1) identify and document the functional and physical characteristics of a configuration item, (2) control changes to those characteristics, (3) record and report change processing and implementation status, and (4) verify compliance with specified requirements [IEEE Std 610, 1990]. (CMMI).
Derived requirements – requirements that are not explicitly stated in the customer requirements, but are inferred (1) from contextual requirements (e.g. applicable standards, laws, policies, common practices and management decisions) or (2) from requirements needed to specify a product component. Derived requirements can also arise during analysis and design of components of the product or system (CMMI).
Documentation – qualifiers include system, environment, code, end-user, help file, training, education, frequently asked questions (FAQ), best practices, etc.
Environment – development, prototyping/staging, implemented
Functional architecture – the hierarchical arrangement of functions, their internal and external functional interfaces and external physical interfaces, their respective functional and performance requirements, and their design constraints (CMMI).
Patch – applies primarily to documentation; not code
Package – A versioned set of tooling software intended to work together to support a stated function with accompanying installation mechanism, user documentation and other support collateral
Product – any tangible output or service that is a result of a process and that is intended for delivery to a customer or end-user (CMMI).
Product requirements – a refinement of the customer requirements into the developer’s
language, making implicit requirements into explicit derived requirements. The developer uses the product requirements to guide the design and building of the product (CMMI).

Project charter – applies to a significant (to be defined) amount of a work; MUST have a project plan; required for the creation of a main code branch or major branch rewrite.

Project manager – the person responsible for planning, directing, controlling, structuring and motivating the project. The project manager is responsible for satisfying the customer (CMMI).

Project plan – a plan that provides the basis for performing and controlling the project’s activities, which addresses the commitments to the project’s customer. Project planning includes estimating the attributes of the work products and tasks, determining the resources needed, negotiating commitments, producing a schedule, and identifying and analyzing project risks. Iterating through these activities may be necessary to establish the project plan (CMMI).

Prototype – a preliminary type, form or instance of a product or product component that serves as a model for later stages of the final, complete version of the product. This model (e.g., physical, electronic, digital, analytical) can be used for the following (and other) purposes:

- assessing the feasibility of a new or unfamiliar technology
- assessing or mitigating technical risk
- validating requirements
- demonstrating critical features
- qualifying a product
- qualifying a process
- characterizing performance or product features
- elucidating physical principles (CCMI).

Requirement – (1) A condition or capability needed by a user to solve a problem or achieve an objective. (2) A condition or capability that must be met or possessed by a product or product component to satisfy a contract, standard, specification, or other formally imposed documents. (3) A documented representation of a condition or capability as in (1) or (2) [IEEE Std 610.12, 1990]. (CMMI).

Requirement analysis – the determination of product-specific performance and functional characteristics based on analyses of customer needs, expectations and constraints; operational concept; projected utilization environments for people, products and processes; measures of effectiveness (CCMI).

Requirements elicitation – using systematic techniques, such as prototypes and structured surveys, to proactively identify and document customer needs and end-user needs (CMMI).

Requirements management – the management of all requirements received by or generated by the project, including both technical and non-technical requirements as well as those requirements levied on the project by the organization (CMMI).

Requirements traceability – the evidence of an association between a requirement and its source requirement, its implementation and its verification (CMMI).

Stakeholders – customers, end-users, suppliers, builders, testers and sponsors

Work product – any artifact produced by a process. These artifacts can include files,
documents, parts of the product, services, processes, specifications and invoices. A key distinction between a work product and a product component is that a work product need not be engineered or be part of the end product (CMMI).