IDEA GROUP PUBLISHING
701 E. Chocolate Avenue, Suite 200, Hershey PA 17033-1240, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITB12860

This paper appears in the book, *Emerging Trends and Challenges in Information Technology Management, Volume 1 and Volume 2* edited by Mehdi Khosrow-Pour © 2006, Idea Group Inc.

HERMES: A XML-Based Environment for Flexible Requirements Management

Alberto Colombo, Ernesto Damiani, & Mauro Madravio
Dept of Information Technology, University of Milan, I-26013 – Crema (CR), Italy, {colombo, damiani, madravio}@dti.unimi.it

Renato Macconi, Research & Development, Tecnologie.net, Software People Group, Italy, renato.macconi@tecnologie.net

Karl Reed, Dept of Science & Computer Engineering, La Trobe University, Bundoora Vic 3083, Australia, kreed@cs.latrobe.edu.au

ABSTRACT

We describe an XML-based approach to requirements engineering that allows flexible tuning of the requirements' structure according to the level attained by an organization in SEI's *Capability Maturity Model* (CMM). As an application, we present ***H**ERMES*, a XML-based, open-source, flexible requirement management environment allowing for tuning requirements' structure according to developers' needs. Our technique is particularly suitable for development in-the-small and lightweight processes like XP[1].

INTRODUCTION

The purpose of traditional requirements' management techniques is producing and maintaining accurate descriptions of customers' software needs. Information representation and organization issues lay at the heart of the requirements management, as a sound organization of requirements can foster their readability and traceability [7].

The need for formal specifications of requirements is due to the fact that requirements can be seen as a contract between software developers and users: therefore they have to be unambiguous and fully comprehensible to consumers.

The Software Engineering Institute (SEI) *Capability Maturity Model* (CMM) states that, in order to qualify for higher maturity levels, organizations should be prepared to enforce a written policy for managing requirements.

Many small-scale developers do not see the benefits of managing requirements as advocated by CMM: their conventional wisdom is that a small developers number guarantees that requirements maintenance can be effortlessly achieved via informal knowledge sharing among customers and programmers.

Our goal is twofold: first of all, we intend to use XML[10] technology to allow system requirements and architectures to be easily shared across the organization[2]. Secondly, we aim to defining a system able to support a more and more sophisticated notion of requirement as companies develop their internal skill and capabilities. Based on its capability of supporting changes, we named our system HERMES, after the Greek god of translators and interpreters.

COMMERCIAL REQUIREMENT MANAGEMENT TOOLS

The HERMES tool is open source, but it has been implemented in order to introduce an accurate requirements management, using a process based on "learning".

The idea of the new RMM scale is to bring users to consider requirements managing as an important and essential method for developing software: requirements are not still considered as a simple manuscript, however also as structured information representing a real requirement document.

Commercial requirements management tools assume a deep understanding of software engineering techniques and other process that are related with, in particular requirements management; on the other hand, if we shall consider a small or medium software-house, we can notice that in a similar context the previous *modus operandi* in not the best one, since there is quite lack in software design.

Try to change the work flow in a little software house could not be so easy and so productive and it could bring none effect none advantage.

Using XML Semi-Structured Data Model to Support CMM Level Transitions

The *Capability Maturity Model* (CMM) [6] is a well-known descriptive model proposed by the Software Engineering Institute (SEI) that provides a framework for organizing software process improvements into five maturity levels.

It should be noted that the CMM model is not prescriptive; it describes an organization at each maturity level without specifying the means for achieving it.

In the sequel, we shall focus on process changes related to the transition form CMM Level 1 to Level 2. This transition deeply affects development practice and methods, primarily by requiring the formalization of all project activities. The requirements gathering and management procedure is among the parts of the software process that are more directly and deeply affected by a transition to CMM Level 2 effort.

The main idea is to represent a requirement as a single entity, transforming it from a text file into an object: to achieve this goal, for example, it possible to insert some mandatory field.

To obtain this actual technologies offer two different ways: implementing requirement as a real object with method, attributes and with inheritance between objects; or using a semi-structured data model, based on metadata to gain this result.

In our opinion, both type support and policies' enforcement can be made easier by passing from a naïve approach to requirements management to a document-centred life-cycle and, finally, to structured data-models; but this transition must be flexible and should not require learning new tools. Also, it is important to be able to support incomplete information in a declarative and reusable form. As we shall see, semi-structured data models such as the *eXtensible Markup Language* (XML) allow for tackling this issue, supporting gradual enrichment of the requirements internal structure while preserving uniform navigation and query interfaces.

The Requirement Management Maturity Scale

Our Requirement Management Maturity (RMM) scale specifies a series of steps intended to help in shepherding small and medium companies toward CMM provisions for Level 2 requirements handling. RMM steps, called levels, do not interfere in any way with CMM ones; rather, they were designed empirically with the help of experienced CMM lead assessors in order to orient and guide the process improvement phase, when consultants gradually shepherd a small company's process from one CMM assessment to the next. As we shall see, RMM levels incrementally define XML formats used for requirements' representation in our system.

Since the CMM has five levels, we transpose each of them into requirement environment, translating proposed concept and strategy for a more specific context; so we propose five different levels:

1. Initial
2. Repeatable
3. Documented
4. Managed
5. Measured

The primary characteristic of five RMM levels is inheritance: each level inherit form previous one, adding new characteristic to improve requirement management process.

This feature, further than to advance organization of each requirement, it could get better the search process: in this way we can recover requirement documents that are not initially stored in same RMM level in which we are in.

USING XML FOR REPRESENTING REQUIREMENTS

The *eXtensible Markup Language* (XML) has become a widely adopted solution for information interchange.

HERMES is centered on single requirement: to enhance in modularity and make the management easier, we use a semi-structured data model that complies with RMM maturity scale propose above. To achieve this, we have to conceive requirement concept from its content: joining two XML-schema we obtain exactly what we are wishing for.

The level 1 (called core) work as general requirement document, more specifically, it defines the root of the HERMES XML format for requirements; instead the others one are designed to reproduce a company-specific requirement structure, therefore allow personalization of requirements; each succeeding level corresponds to a RMM level transition, so that the semantics of additional portions of the requirement can emerge at each step.

You have to notice that each schema-level corresponds to a RMM level transition, so that the semantics of additional portions of the requirement can emerge at each step.

When a requirement is validated against the core level schema, its XML mark-up is not expected to carry any information; all its semantics is expressed by the natural language terms contained in the attribute value, exactly like in conventional document-centred environments and the requirements' repository internal structure is *flat*.

This type of multilevel schema representation of single requirement is very simple, but at the same time is powerful: upgrading from a level to next one (for example from RMM level 1 to RMM level 2), we keep information in all requirements previously stored, simply integrating new feature required at this point.

From RMM level 2 we introduce components administration: in this case user can develop software module and link them one or more requirements using *OwnerComponent* element; we also introduce information about test case: it specify the location leading to the set of test cases relevant for testing the requirement [4, 2].

For linking requirement to a component we adopt DOI (Digital Object Identifier) system [3].

DOI system is a set of names (characters and/or digits) assigned to a digital object in order to identify it clearly in Internet. The information can be changed, but its DOI will not be modified.

In this schema there is also a *Key* field; this element is hidden and that indicates which fields are used for sorting and grouping requirements in version tree (see paragraph later): for example it could contains author name and date.

From RMM level 3 it's also managed requirements versioning: it is based on CVS (Concurrent Versions System) system.

Versioning is treated as joining two filed: major number and minor number; user must also justify the requirement change.

If user modifies some sensible field (for example author, requirement's name or other filed user wants to set), the system generates automatically a new version of the requirement, modifying minor number of requirement.

User is also able to upgrade version of a requirement, producing a new one with newest major number; if user wants to modify a requirement that is yet connected to a component, he has to make a new version of requirement and not evolutes it.

THE HERMES SYSTEM ARCHITECTURE

HERMES is a collaborative, distributed requirement management system built around a XML repository that stores both requirement documents and our multi-level schema definitions.

The HERMES repository can either exploit native XML storage or rely on a conventional RDBMS; in both cases, it is accessed via a Web interface allowing for updating, searching and querying the requirements base.

LinkManager is a module based on DOI systems: it's used to obtain a unique number fro a resource; this module could be a real DOI server, so number are really generate by DOI system in internet; if company do not need a real DOI server, DOI number is generated by our module, and it's unique inside HERMES environment. If user links a requirement to a component, system makes a request to LinkManager module in order to obtain a DOI number.

Using HERMES with Agile Processes

An interesting feature of our environment is that it is completely independent from the software process. Process independence is very important because many of the small companies we targeted when developing our tool use lightweight processes or no process at all.

HERMES can easily represent artefacts relevant to agile methodologies, supporting sharing and modification. As an example, here we briefly deal with the *eXtreme Programming* (XP) approach[1,8,9]. XP does not include specific provisions for a requirements lifecycle. The XP concept that comes closest to a requirement is the one of *User Story* (US) summarizing scenarios of users' interaction with a system.

CONCLUSIONS

It is widely acknowledged that requirements management may improve software quality and reduce life-cycle time; in organizations at a high maturity level, structured requirements can result in a better software design and more maintainable code.

However, this is not the case for small companies at a low maturity level. Specifically, transition from CMM Level 1 to Level 2 must be accompanied by flexible, gradual enrichment of the requirements structure and by a smooth increase of requirement management tools' functionalities. This paper presented an environment based on state-of-the-art XML schemata for requirements modelling. Experience with small and medium enterprises in the software industry suggests that introducing new data model features of requirements should be as strictly related as possible to fine-grained steps taken in improving the software process.

It is possible to download a version of HERMES system visiting http://ra.crema.unimi.it/hermes.

REFERENCES

Kent Beck, Cynthia Andres. Extreme Programming Explained : Embrace Change,Addison-Wesley 2nd (2004)

Rex Black, Rex Black. Managing the Testing Process: Practical Tools and Techniques for Managing Hardware and Software Testing. Wiley, 2nd Edition (2002)

The DOI Handbook, International DOI Foundation Inc,2005, http://www.doi.org/hb.html

Glenford J. Myers, Corey Sandler, Tom Badgett, Todd M. Thomas.The Art of Software Testing. Wiley, 2nd Edition (2004)

Jun Han, Jie Wu "XmlTRAM+: Using XML Technology to Manage Software Requirements and Architectures", AusWeb 2002, July 2002.

J. Herbsleb, D. Zubrow, D. Goldenson, W. Hayes, M. Paul, Software quality and the Capability Maturity Model, Communications of the ACM 40(6), (1997)

D. Leffingwell, D. Widrig, Managing Software Requirements, Addison-Wesley, 2000. [Mac97] L. Macaulay, Requirements Engineering, Springer Verlag, 1997

G. Succi, J. Yip, E. Liu Analysis of the essential requirements for a domain analysis tool, Proceedings of ICSE Workshop on Software Product Lines Economics, 2000

G. Succi, M. Marchesi. Extreme Programming Examined, Pearson Education (2001)

W3C. Extensible Markup Language (XML) 1.0. Feb. 1998. http://wB.w3C.org/TH/KEC-xml/ [XSLT] W3C. XSL Transformations (XSLT) Version 1.0 W3C Recommendation 16 November 1999 http://www.v3.org/TR/xsIt

ENDNOTES

1 This was supported by MAPS project within the FIRB research programme of the Italian Research Ministry

2 Of course, this is the basic benefit that XML is supposed to bring to document management in general. However, commercial requirement management environments are being quite slow in adding even straightforward XML-based representation. The paper [5] was among the first to present a DOOR-based requirement management system providing basic XML capabilities.