SEKE 2010

Proceedings of the Twenty-Second International Conference on Software Engineering & Knowledge Engineering

San Francisco Bay
July 1-3, 2010

PROCEEDINGS

SEKE 2010

The 22nd International Conference on Software Engineering & Knowledge Engineering

Sponsored by

Knowledge Systems Institute Graduate School, USA

Technical Program

July 1-3, 2010
Hotel Sofitel, Redwood City, San Francisco Bay, USA

Organized by

Knowledge Systems Institute Graduate School

ISBN-10: 1-891706-26-8 (paper)
ISBN-13: 978-1-891706-26-4

Additional Copies can be ordered from:
Knowledge Systems Institute Graduate School
3420 Main Street
Skokie, IL 60076, USA
Tel:+1-847-679-3135
Fax:+1-847-679-3166
Email:office@ksi.edu
http://www.ksi.edu

Proceedings preparation, editing and printing are sponsored by
Knowledge Systems Institute Graduate School

Printed by Knowledge Systems Institute Graduate School

SEKE2010 Foreword

On behalf of the Program Committee Co-Chairs and the Program Committee of the 2010 International Conference on Software Engineering and Knowledge Engineering (SEKE2010), we sincerely welcome you to contribute and attend SEKE-2010 in San Francisco, California, USA.

The International Conference on Software Engineering and Knowledge Engineering has entered its 22nd year. In the past twenty-one years, the International Conference on Software Engineering and Knowledge Engineering has provided a unique and important forum for academic and industrial researchers and practitioners to exchange research ideas, results and application experience in software engineering and knowledge engineering.

This year's technical program is prepared and organized by a great team of Program Co- Chairs, who are listed below.

Program Co-Chairs:

- Marek Reformat, University of Alberta, Canada
- S. Masoud Sadjadi, Florida International University, USA
- Du Zhang, California State University Sacramento, USA

It has been my great honor and pleasure to serve as the SEKE2010 Program Committee Chair and work with this great team and the program committee members to prepare a rich and solid technical program as well as the high quality conference proceedings. The published conference proceedings contain the papers accepted and selected for presentation at SEKE2010 based on a rigorous review process. I hope it will serve as a valuable reference for the research community in the coming years.

We received an overwhelming 253 submissions from 32 countries. The acceptance rate for full papers is 33%, and that for short papers is 29%. This year, authors from 32 countries (Algeria, Argentina, Australia, Austria, Brazil, Canada, China, Colombia, Egypt, France, Germany, India, Iran, Israel, Italy, Japan, Jordan, Malaysia, Mexico, Morocco, Netherlands, Poland, Saudi Arabia, South Korea, Spain, Sweden, Switzerland, Taiwan, Tunisia, Turkey, United Kingdom, and United States) will present papers at the conference.

As the Program Chair for this Conference, I am very grateful to have this opportunity to work with three distinguished SEKE2010 Program Committee Co-Chairs and the committed program committee members. Their excellent support and prompt review efforts led to the successful organization of SEKE2010 technical program. I want to extend my sincere and deepest thanks to Dr. Xiaoying Bai, Dr. Shihong Huang as the Publicity Co-Chairs, Dr. Hironori Washizaki as the Asia Liaison, Dr. Jose Carlos Maldonado as the South America Liaison. My appreciation also goes to the keynote speakers and special address presenter for sharing their visions, insights, and experiences with the conference attendee about emergent technologies and trends, research topics and issues in both academic research and industry applications. Moreover, I like to express my appreciation to the organizers of the workshops and special sessions for their great contributions and supporting efforts to make SEKE2010 a great success. In addition, I would like to thank Dr. S. K. Chang, the Steering Committee Chair, Dr. Guido Wirtz, the Conference Chair, and Dr. Guenther Ruhe, the Honorary Conference Chair, for their excellent guidance throughout the conference preparation process. Last but not the least; we owe a special gratitude to Mr. Daniel Li and Mr. Alec Liu from the Knowledge Systems Institute for their great efforts and timely support.

Finally, I truly hope you will enjoy the technical program of SEKE2010, have productive discussion, great presentation and networking. Of course, I sincerely hope you will all explore and enjoy the unique summer weather and various attractions in San Francisco area.

Jerry Gao
SEKE2010 Program Chair

The 22nd International Conference on Software Engineering & Knowledge Engineering (SEKE 2010)

July 1-3, 2010
Hotel Sofitel, Redwood City, San Francisco Bay, USA

Conference Organization

Steering Committee Chair

Shi-Kuo Chang, *University of Pittsburgh, USA*

Steering Committee

Vic Basili, *University of Maryland, USA*
Bruce Buchanan, *University of Pittsburgh, USA*
C. V. Ramamoorthy, *University of California, Berkeley, USA*

Conference Chair

Guido Wirtz, *Bamberg University, Germany*

Honorary Conference Chair

Guenther Ruhe, *University of Calgary, Canada*

Program Chair

Jerry Gao, *San Jose State University, USA*

Table of Contents

Software Requirement Engineering

Software Validation

Interoperability and Semantic Web technologies

Software Quality Assurance

Software Measurement

Software Architectures

Software Test Automation, Practice, and Standardization

Software Engineering with Computational Intelligence and Machine Learning

Web Services

Software Maintenance and Evolution

Human-Computer Interface and Interaction

Empirical Software Engineering and Software Economics

Service-Oriented Architectures and Applications

Software Test Automation and Experience

Knowledge Engineering Systems

Software Project Management

Software Engineering Tools and Experience Report

Agent-Based Software Engineering and Applications

Component-Based Software Engineering

Supporting Software Process Improvement in Very Small Entities through a Template-based Guide

Miguel Morales Trujillo,
Guadalupe E. Ibargüengoitia
Graduate Science and Engineering Computing, UNAM
Mexico City, Mexico
{migmor, gig}@ciencias.unam.mx

Francisco J. Pino
IDIS Research Group – Electronic and Telecommunications Engineering Faculty.
University of Cauca
Popayán, Colombia
fjpino@unicauca.edu.co

Mario Piattini
Alarcos Research Group – Institute of Information Technologies & Systems.
University of Castilla-La Mancha
Ciudad Real, Spain
Mario.Piattini@uclm.es

Abstract— **When a VSE decides to implement a software process model, the first step is to choose a reference model that is appropriate to its working environment and possibilities, which must then be adjusted to its specific needs. We believe that it would be simpler to adapt the reference model to the needs of the VSE through the use of templates, which would permit the correct integration of the activities proposed in the model and of those that the company has already been carrying out, thus narrowing the gap between the chosen model and its adoption. This paper presents a template-based guide as an alternative through which to implement a process reference model. The templates are developed to capture and concentrate the results of the activities stated in the reference model, thus creating standardized products that meet the appropriate requirements needed to increase the companies' maturity and capability. In accordance with the activities recommended in the COMPETISOFT reference model, we propose templates as an alternative means of generating a product associated with each activity. The templates are made up of individual items - atomic template-units - each of which is supported by the Unified Process and PMBoK methodologies. Our intention is to incorporate the processes into the VSEs, with the support of templates that are manageable, easy and understandable, taking into account the inherent limitations of small businesses.**

Keywords: Template-based, VSE, software process model, software process improvement, process reference model, COMPETISOFT, ISO/IEC PDTR 29110

I. INTRODUCTION

Software Process Improvement (SPI) is a controlled and guided effort whose objective is to improve process sets in a software development organization, and particularly its capability and maturity. All initiatives to implement an SPI must follow a methodological framework that will provide certain activities leading to the desired improvement.

Very small entities (VSEs), which consist of less than 25 employees, are fundamental to a country's economy. In countries such as the US, Brazil, Canada, China, India, Finland, Ireland, or Hungary, small companies represent up to 85 percent of all software organizations [1] [2].

Nevertheless, if VSEs are to constantly develop and improve, they must adopt software process reference models. However, there is a widely held belief that process reference models are too expensive, require a lot of time and are intended for larger enterprises (more than 100 employees) thus making them difficult for VSEs to acquire [6] [7].

According to [20] other reasons for this phenomenon include: the VSEs remain unaware of these methodologies; in addition, the proposals are difficult to apply owing to the large investment of time, money and resources involved in an SPI. What is more, the proposals are not suitable for the particular characteristics of this type of companies.

In an effort to help VSEs adopt processes in everyday work, process reference models such as COMPETISOFT [3] [4] and ISO/IEC PDTR 29110 [5] were created. These models simplify the required tasks and help to achieve the software engineering practices correctly. However, complex issues still remain and a considerable amount of time is needed to assimilate them.

At the moment of capturing the results of the activities executed, the organization confronts the difficulty of how to represent them in a tangible product. An alternative to the implementation of SPI, in accordance with the process reference models, is a template-based proposal.

The use of a set of templates makes it easier to adopt a reference model [13]. In support of this strategy, this paper describes a template-based guide for SPI implementation in accordance with COMPETISOFT. We propose templates as an alternative means of generating a product associated with each activity recommended by the process reference model.

The use of a template helps to reduce the gap in the knowledge of models, since the template is ready to capture the results of the activities to be executed, and there is no need to resort to any other additional source of knowledge. This signifies that the template acts as an explicit guide and a result container for each activity, always following the required regulations.

This paper is organized as follows: the background is presented in Section II, Section III demonstrates the templates and their characteristics, Section IV describes two case studies and the final section discusses conclusions and future work.

II. BACKGROUND

Reference models such as COMPETISOFT and ISO/IEC PDTR 29110 are already available for VSEs. They define groups of processes, which later group the activities needed to achieve the desirable improvement. However, these models tend to focus on merely stating the activities and products that are generated by their implementation. We offer a template-based approach which follows the COMPETISOFT framework, in which the template guides the activities and captures their results to generate standard and regulation-oriented products that will boost an organization's maturity and process capability.

COMPETISOFT emerged with the objective of increasing competitiveness among small Latin-American software enterprises, along with providing a common framework that would be easily adjusted to the specific needs of any small enterprise with the purpose of establishing an evaluation and certification mechanism that operates specifically for VSEs.

By seeking to adapt the model to the organization rather than vice versa, COMPETISOFT consistently integrates basic and essential elements in order to ease its implementation. The reference model is divided into three levels: Top Management, Management and Operations, thus reflecting the basic organization division of a VSE. Top Management is consequently in charge of strategic planning and continuous review within the organization; Management is responsible for providing resources, processes and projects, together with supervising the accomplishment of the organization's goals; and finally, Operations is where Project Management, Software Development and Maintenance processes are concentrated.

The intention of the Operations level is to ensure that every software development project that the organization is running is accomplished within the constraints of time, cost and quality [17]. The processes responsible for accomplishing these objectives are: Project Management, the person in charge of finishing the processes on time and within the estimated cost; Software development, which ensures that the product meets the requirements and eliminates defects during early stages of development through verification and validation, and Software Maintenance, which is required after the product is released. Owing to the vital role that the Operations level plays in an organization, this paper focuses on introducing the template-based SPI for these groups of processes.

Having defined the scope of our template-based proposal, we then established an object oriented methodology to control the templates' structure and development. In this case the templates are based on the Unified Process (henceforth referred to as UP) [8] [9] and the Project Management Body of Knowledge (henceforth referred to as PMBoK) [10], since these regulations are applicable to any software development process.

We chose the UP to guide and control the templates since, owing to its similarity with COMPETISOFT, both propose an iterative and incremental development and are highly adaptable to any organization's specific needs, thus motivating the achievement of high quality standards. Moreover, the fact that UP is accepted as a good reference by the software industry makes it an excellent alternative for the templates' objectives.

As for PMBoK, this is a clearly defined standard for project management, which falls into nine knowledge areas and embraces the best practices, in which the information referring to project administration is standardized in a consistent manner, and that knowledge is then applied to software engineering.

In the following section of the paper, we present the characteristics and development of the templates that are expected to be an explicit guide towards how to make process improvement activities and their fulfillment feasible.

III. TEMPLATES AND THEIR CHARACTERISTICS

The use of templates as process improvement guides seeks to make a significant contribution to an organization by using them as a tool [18] [19] to reduce the gap between the VSE and the reference model.

The development of the templates began with those that would be used in the Project Management processes, followed by those that would be used in Development and Maintenance. Product generating and non-generating activities were identified for each of the processes. For those activities that have an associated product, the respective templates provide a technique to accomplish them. For example, our proposal is to represent the capture of requirements by means of a Use-Case Diagram; a *requirements workshop* is suggested to create a user interface prototype, and the use of a *dependency graph* is proposed to develop the integration test plan.

In order to bring the activities to a successful conclusion and to achieve their goals, we chose to extract good practices from PMBoK for the Project Management and from the UP for the Software Development and Maintenance processes. A mapping between RUP and PMBoK is presented in [11]; this mapping proves that there are no contradictions or fundamental incompatibilities between the two standards, and that their practices can therefore coexist without abnormalities. It is thus possible to correlate both the PMBoK and the UP guidelines.

The decision to choose UP for Software Development and Maintenance is mostly owing to the fact that those processes are guided by use-cases and they focus on the architecture. In addition, and according to the conclusions reached in the research conducted in [12], the scope of coverage between those two processes is largely attained in accordance with the ISO/IEC 15504 qualification [15]. The ISO/IEC 15504 regulation provides a model for process evaluation and the coverage scope is determined by establishing the minimum requirements that the processes must fulfill.

However, we discovered that some of the products suggested by COMPETISOFT are not taken into account by the UP, for instance, Integration Test Plan and Report. In COMPETISOFT an entire development phase is designated for this activity. For that reason, in some cases we opted to

follow industry examples that were adapted to the objectives of this project.

The main feature of the templates is the modularity of the entities of which they are composed. These entities can be divided into two types: general entities and process entities. The general entities are common to all the templates and focus on describing the following aspects: Introduction, Overall description, Rationale for decisions and Supporting information. The process entities will capture all the information generated by the activities described in the reference model.

These entities will be referred to as *atomic template units*, and each *atomic unit* contains a technique with which to generate its content. Altogether, they represent a detailed picture of the step-by-step creation of a product that would meet all the requirements and would fit in suitably with the rest of the products. The division of templates into atomic units facilitates traceability of the software elements during their life-cycle. An example of this could be the trace connecting client's requirements, use-cases and classes in order to implement them, where each component can be easily traced and associated at any point of development.

Every process component is chosen in accordance with COMPETISOFT requirements, always ensuring that the template is simple and self-contained. The Testing template can serve as an example, in which the atomic unit Testing Strategy is responsible for the tests that are not considered by the methodology, but are required by the organization. Each template is assisted by an *Atomic Unit Diagram* which serves as a guide for its correct use.

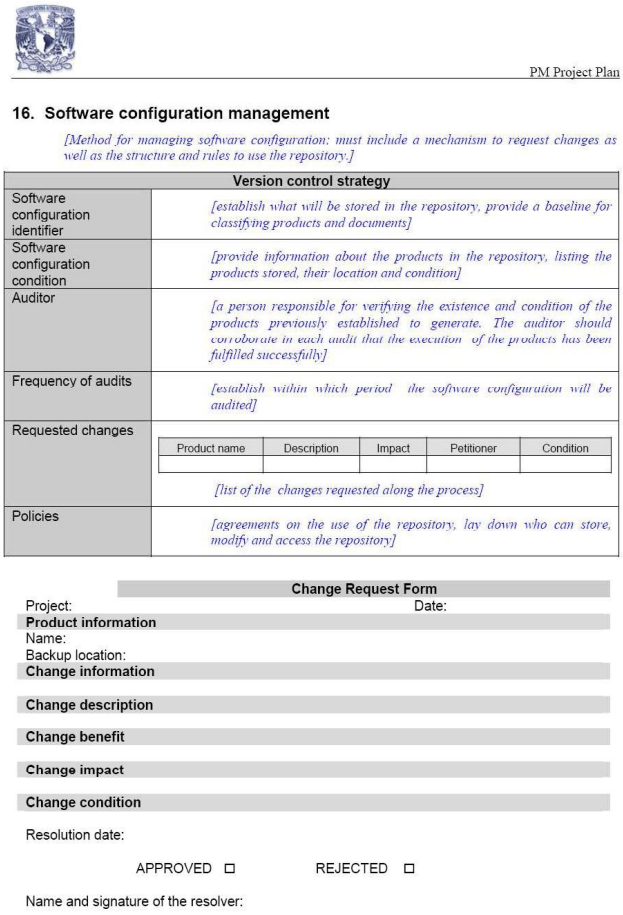
To summarize, one of the key features of the templates is the modularity of their process entities which allows them to be easily adapted to different reference models.

A. Project Management Templates

The first process subgroup concerns the Project Management (PM), which is in charge of the establishment and systematic fulfillment of the activities needed to achieve the time and cost goals of a project. The Project Management activities produced four templates; three of these are products that are repeatedly generated during the process: *Traceability Matrix*, *Acceptance Document* and *Monitoring Report*. These documents were therefore standardized and included those specific elements from each activity that are considered to be good practices by PMBoK. The fourth template, Project Plan, describes the information related to the project, which includes and records PMBoK techniques.

It is worth pointing out that we attempted to make the filling out of the Project Management templates as intuitive and natural as possible, since these templates are used several times throughout the process.

Figure 1 shows the atomic unit Version Control Strategy which, in accordance with PMBoK techniques, defines the way in which the changes will be managed during the development, maintaining the baselines' integrity. Moreover, another part of the Project Plan template is the Change Request Form, which is essential for keeping a record of project changes.



PM Project Plan

16. Software configuration management

[Method for managing software configuration; must include a mechanism to request changes as well as the structure and rules to use the repository.]

Version control strategy	
Software configuration identifier	*[establish what will be stored in the repository, provide a baseline for classifying products and documents]*
Software configuration condition	*[provide information about the products in the repository, listing the products stored, their location and condition]*
Auditor	*[a person responsible for verifying the existence and condition of the products previously established to generate. The auditor should corroborate in each audit that the execution of the products has been fulfilled successfully]*
Frequency of audits	*[establish within which period the software configuration will be audited]*
Requested changes	Product name | Description | Impact | Petitioner | Condition *[list of the changes requested along the process]*
Policies	*[agreements on the use of the repository, lay down who can store, modify and access the repository]*

Change Request Form

Project: Date:

Product information

Name:

Backup location:

Change information

Change description

Change benefit

Change impact

Change condition

Resolution date:

APPROVED ☐ REJECTED ☐

Name and signature of the resolver:

Figure 1. PM Project Plan template fragment.

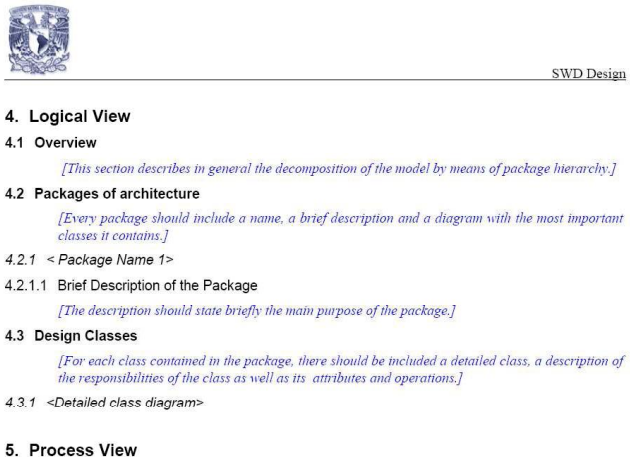
B. Software Development Templates

The products required by Software Development (SWD) processes are incorporated in one template for each of the phases proposed by COMPETISOFT. This produces seven elements, of which the *Requirement Specification* comes first. This template includes the atomic unit specially designed for modeling the requirements by means of use-case technique, this atomic unit being the most important part of the document. The life-cycle traceability of each element begins with the registration of every requirement. The System Testing Plan and the User Interface Prototype should be included in the same document to cover every use-case.

The *Analysis*, *Design*, *Construction*, *Integration*, *Test* and *Closure* templates were also designed.

As an example of these templates in Figure 2, we can observe the elements that make up the Logical View (point 4), where a description of the view (4.1), along with the definition of the architecture (4.2), is provided by using packages. The Deployment View (6) is where a deployment diagram (6.2) needs to be provided for its presentation. Finally, the Data View (7.) includes the instructions necessary to define aspects concerning data persistence (7.1.) and represent them in a diagram (7.1.1.). All of the above is based on and adapted from the UP Design Phase techniques.

It is necessary to mention that the atomic units could be reorganized into subsets to meet the needs of other reference models.



SWD Design

4. Logical View

4.1 Overview

[This section describes in general the decomposition of the model by means of package hierarchy.]

4.2 Packages of architecture

[Every package should include a name, a brief description and a diagram with the most important classes it contains.]

4.2.1 < Package Name 1>

4.2.1.1 Brief Description of the Package

[The description should state briefly the main purpose of the package.]

4.3 Design Classes

[For each class contained in the package, there should be included a detailed class, a description of the responsibilities of the class as well as its attributes and operations.]

4.3.1 <Detailed class diagram>

5. Process View

5.1 Overview

[This section describes the system decomposition into processes and process groups, organized according to the interactions between them. It describes the principal means of communication used in these interactions.]

6. Deployment View

6.1 Overview

[This section should describe the software network configuration for its execution. It should indicate the nodes (computers, routers, etc.) necessary for the proper functioning of the software, and interconnections between nodes (bus, LAN, etc.) Ideally it should include a mechanism or protocol of node communication.]

6.2 Deployment model

[Distribution of the identified nodes.]

7. Data View

7.1 Overview

[This section should include a description of the data storage mechanism of the system. This section is optional in case if there is little or no persistent data or the translation between the design model and the data model is trivial.]

7.1.1 <Database Diagram >

Figure 2. SWD Design template fragment.

C. *Software Maintenance Templates*

The Software Maintenance (SWM) process established by COMPETISOFT is an agile process [16]. In order to carry it out successfully, we abstracted two necessary elements: the *Modification Request* template, from which the maintenance team participation is requested, and the *Intervention* template in which a detailed description necessary to perform the requested modification is provided.

These templates have the same structure as the Project Management templates, and are meant to be user-friendly, owing to the fact that the *Modification Request* could be filled out by a system user and not by a software developer, as is shown in Figure 3. In this figure, we can observe that each entity of the template appears to be easy to abstract and that the information obtained would be useful for the maintenance team. Each atomic unit represents an explicit guide consisting of direct questions for the template user, who must give an immediate, clear and sufficiently complete answer.

SWM Modification Request

Modification Request

Priority	*[Normal / Urgent]*
Error/Change description	*[This section should answer questions like these:* *Who experienced the error?* *When did the error occur?* *How did the error occur?* *These questions must be answered by the system user who gets the error occurred. The purpose of this section is to determine actions that triggered the error.]*
Occurrence environment	*[This section should include a list of the features presented in the system environment at the time of the error submission.]*
Affected aspects	*[This section should list the issues that are affected by the described defect or the occurrence of the error.]*
Requested by	*[Name of the person making the request, in case of being other than the user who experienced the error include the name of the user.]*
Diagnosis and potential solutions	*[In this section possible solutions to the error should be proposed as well as error diagnosis, ie establishing the nature of the error by observing the signals causing it.]*

Figure 3. SWM Modification Request template.

D. *Atomic Unit Diagrams*

In order to provide a graphic representation of the implementation of the templates during the process, diagrams were generated, in which three important elements were conjoined: roles involved, list of activities described in the process reference model and the template's atomic units in-use. Figure 4 presents the atomic unit diagram related to the Project Plan template. On the left-hand side of the figure there is a graph showing the activities described in the Project Management process taken from the COMPETISOFT reference model. (Sequence A. 1.)

In this diagram, the roles (columns) are related to each activity by means of swim lanes. The template's atomic unit associated to a corresponding activity of the process is positioned in each lane, therefore resulting in a clear and consistent mapping between the process reference model and the templates. The objective of the aforementioned diagram is that of meeting specific needs of the activities.

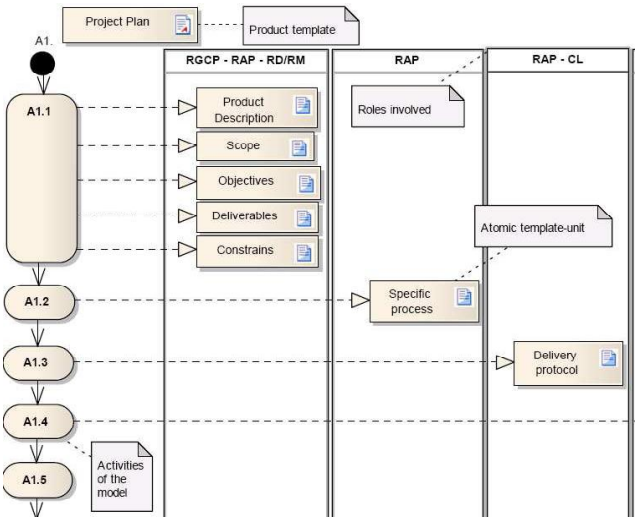


Figure 4. PM Project Plan Atomic Unit Diagram fragment

E. *Template Adjustment to ISO/IEC 29110-5-1*

Taking advantage of the modular structure of the templates, a set of atomic units was chosen to be adjusted to ISO/IEC 29110-5-1, with the objective of designing templates that would be part of each deployment package [14] proposed by this future regulation.

Figure 5 shows the content that resulted from gathering together suitable atomic units in order to complete the Software Requirements Analysis deployment package from ISO/IEC 29110-5-1. This quick and easy adjustment to the deployment package is an example of the benefits of choosing to structure the templates from atomic units. The modular aspect of templates is meant to make process implementation more flexible and user-friendly.

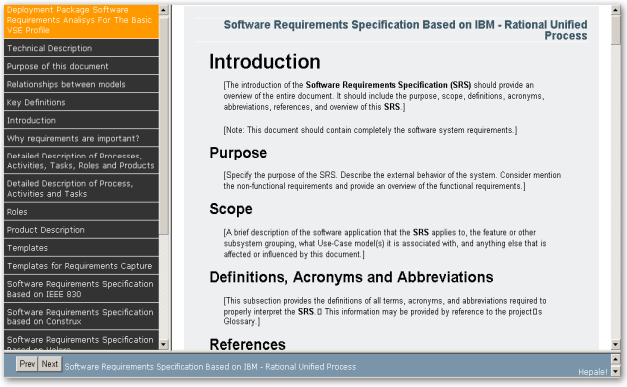


Figure 5. Content of Software Requirements Analysis deployment package template.

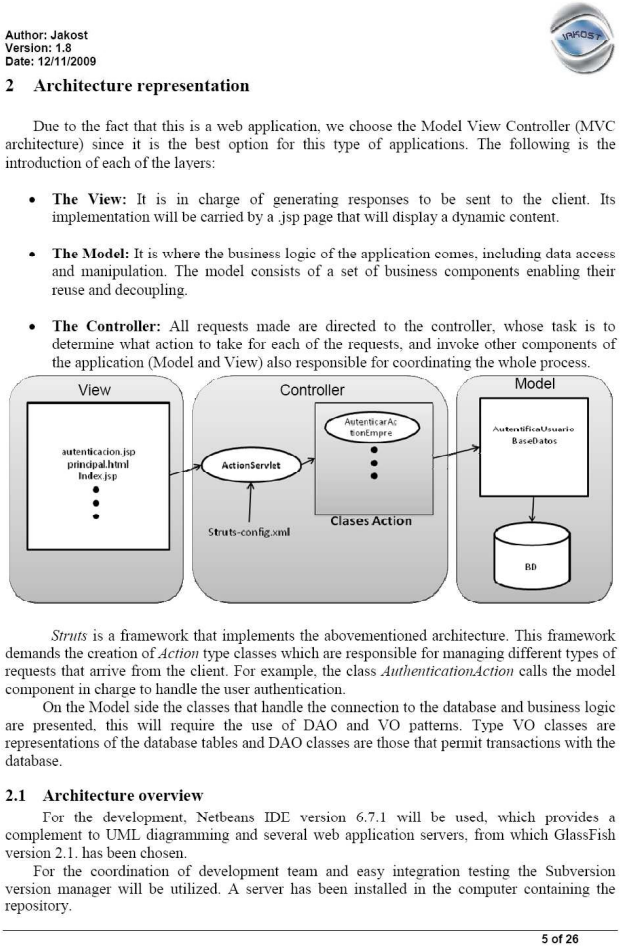
IV. CASES STUDY

The templates designed for the Project Management and Software Development processes were used in the development of three software products related to an academic environment. The first case is the application of templates to a tool development project called HEPALE! (Educational Tool for the Adoption of Standards from its acronym in Spanish) [21], which consists of two modules, both of which are in charge of integrating distance learning standards to spread COMPETISOFT and ISO/IEC 29110 models. Moreover, this tool obtains useful statistics which can be analyzed to provide means to improve educational offerings and to contribute to the dissemination of reference models at an international level. The other use of templates was conducted during the creation of a repository, which contains all the changes needed to manipulate learning objects and e-learning principles in its structure. The architecture of the repository is service-oriented and implements web services to manipulate, search and store deployment packages. The repository not only makes the interchange of contents among different Learning Administration Systems easier, but also facilitates their publication.

The HEMOSIST case study was developed outside the academic environment and with a real client. This system was designed for the National Institute of Cardiology to manage the database records of Hemodynamic patients. In order for the system to be easily upgraded, there was an essential need for proper documentation and for a software process reference model that could be followed.

The development team was composed of 5 people who had never followed a model and who adopted the template-based proposal to guide their development according to COMPETISOFT and to meet the quality requirements. The team confirmed that the application of templates assisted in following the model, reducing the effort and time to adopt it, which permitted them to focus on the development of the system to a greater extent. At the end of the development, the system was delivered on time and with the quality standard required, whilst the organization obtained a full system documentation, leaving it ready for an upgrade or maintenance process. At present the system is successfully in use at the Institute.

As an example of the use of the templates, the document resulting from the implementation of the Analysis template during the development of HEMOSIST is shown as follows. Figure 6 demonstrates the outcome of applying the techniques proposed for the Architecture Representation atomic unit, and a diagram, goals and system architecture restrictions are presented.


Author: Jakost
Version: 1.8
Date: 12/11/2009

2 Architecture representation

Due to the fact that this is a web application, we choose the Model View Controller (MVC architecture) since it is the best option for this type of applications. The following is the introduction of each of the layers:

- **The View:** It is in charge of generating responses to be sent to the client. Its implementation will be carried by a .jsp page that will display a dynamic content.
- **The Model:** It is where the business logic of the application comes, including data access and manipulation. The model consists of a set of business components enabling their reuse and decoupling.
- **The Controller:** All requests made are directed to the controller, whose task is to determine what action to take for each of the requests, and invoke other components of the application (Model and View) also responsible for coordinating the whole process.

Struts is a framework that implements the abovementioned architecture. This framework demands the creation of *Action* type classes which are responsible for managing different types of requests that arrive from the client. For example, the class *AuthenticationAction* calls the model component in charge to handle the user authentication.

On the Model side the classes that handle the connection to the database and business logic are presented, this will require the use of DAO and VO patterns. Type VO classes are representations of the database tables and DAO classes are those that permit transactions with the database.

2.1 Architecture overview

For the development, Netbeans IDE version 6.7.1 will be used, which provides a complement to UML diagramming and several web application servers, from which GlassFish version 2.1. has been chosen.

For the coordination of development team and easy integration testing the Subversion version manager will be utilized. A server has been installed in the computer containing the repository.

5 of 26

Figure 6. Example of use of the Architecture Representation atomic unit.

V. CONCLUSIONS AND FUTURE WORK

Approximately 80% of software development organizations are categorized as VSEs, that is, they have 25 or less employees. This overwhelming majority leads to the necessity for a reference model which can be adapted to VSEs rather than vice versa. Bearing in mind the aforementioned aspects, a template-based guide was proposed to assist in the adoption of reference models with the aim of ensuring process improvement without forgetting the organizations' real working environment and real needs. In this paper we have presented the template-based guide designed to introduce and follow COMPETISOFT. Among the guide's characteristics are flexibility and ease-of-adjustment to the organization's real needs provided by the modular structure of templates. Moreover, these templates can be adapted and used with other reference models, such as ISO/IEC CD TR 29110-5-1, thanks to the use of process entities in atomic units.

Some of the positive comments and aspects mentioned by the people who have used the templates are: (i) The templates facilitate and show the best course to follow when carrying out process activities according to a reference model; (ii) The templates include all the aspects that are necessary and are required by a reference model, and it is therefore difficult to overlook any of them; (iii) The templates serve as fill-in forms and incorporate all the resulting information, which allows a complete, organized and standardized documentation to be obtained at the end of a process; (iv) The use of atomic units ensures the traceability of elements and gives the template remarkable flexibility and usefulness since the developer is free to decide whether or not to use each atomic unit; and (v) The way in which the atomic units are constructed, and particularly those of Project Management and Software Maintenance, reduces the time and effort needed to fill out forms, making it simple and easy for any user.

As future work we shall consider the following: (i) To integrate the template-based proposal into all previous work done with regard to COMPETISOFT, thus making the templates accessible and available for use to any software organization; (ii) To obtain a larger number of applications of the templates in order to evaluate, improve and refine their content; and (iii) To continue a widespread industrial implementation in real software development organizations.

ACKNOWLEDGMENTS

This work has been funded by the following projects: ARMONÍAS (JCCM of Spain, PII2I09-0223-7948), INGENIO (JCCM of Spain, PAC08-0154-9262), PEGASO/MAGO (MICINN and FEDER of Spain, TIN2009-13718-C02-01), PAEP and the mixed-grant scholarship program of CONACYT.

REFERENCES

[1] Qualidade no Setor de Software Brasileiro-Pesquisa 2001, Ministry of Science and Technology (MTC), 2001; http://ctjovem.mct.gov.br/index.php/content/view/34854.htm 20/02/2010.

[2] Software Industry Statics for 1991-2005, Enterprise Ireland, 2006; www.nsd.ie/htm/ssii/stat.htm, 15/01/2009.

[3] Oktaba, H.; García, F., Piattini, M., Pino, F., Alquicira, C., Ruíz, F. "Software Process Improvement: The COMPETISOFT Project". IEEE Computer, October, 2007. Vol. 40(10), pp. 21-28.

[4] COMPETISOFT, http://alarcos.inf-cr.uclm.es/Competisoft/ ,20/02/2010.

[5] ISO/IEC 29110 Draft version, BER-005 29110 TR VSEP Overview WD2 15CL.

[6] Richardson Ita, Gresse von Wangenheim Christiane, "Why Are Small Software Organizations Different?", IEEE Software, January/February 2007. Vol. (40). pp.18-20.

[7] Laporte, C. Y., Alexandre, S., O'Connor, R., "A software engineering lifecycle standard for very small enterprises", in R. V. O'Connor et al. (Eds.): EuroSPI 2008, CCIS 16, pp. 129-141

[8] Kruchten P., The Rational Unified Process. An Introduction. 2nd edition. Addison Wesley. 2000."

[9] Jacobson I., Booch G., Rumbaugh J., The Unified Software Development Process, Pearson, 2000

[10] A Guide to the Project Management Body of Knowledge, Third Edition, Project Management Institute, USA, 2004

[11] Charbonneau, S. "A mapping between RUP and the PMBok", http://www.ibm.com/developerworks/rational/library/4721.html#author, 01/05/2009

[12] Cánepa K., Dávila, A., "Mapeo de los Procesos de RUP respecto a MoProSoft", JIISIC 08, pp. 139-146

[13] Kong, B., Luo, X., Jiang, Y., Mao, M. "Template-based software process improvement", Journal of Information and Computational Science 1 (2), pp. 175-180

[14] Laporte, C., Alexandre, S., Renault, A., "Developing International Standards for Very Small Entities", IEEE Computer 41(3): 98-101 (2008).

[15] van Loon, H. (2007a) "Process Assessment and ISO 15504", Springer.

[16] Pino, F., Triñanes, G., García, F. y Piattini, M. "Agil_MANTEMA: Una metodología de mantenimiento de software para pequeñas organizaciones". JISBD 08, pp.171-182

[17] Oktaba, H., Piattini, M., Pino, F., Orozco, M. Julia., Alquicira, C., COMPETISOFT: Mejora de Procesos de Software para Pequeñas y Medianas Empresas y Proyectos, Ra-Ma, 2008

[18] Word Template Index, http://homepages.fh-giessen.de/~hg14540/SQM/projekt/rup/wordtmpl/index.htm

[19] Rational Unified Process: Artifact Overview http://www.ts.mah.se/RUP/RationalUnifiedProcess/process/artifact/ovu_arts.htm

[20] Pino, F., Hurtado, J., Vidal, J., García, F., Piattini, M. "A process driving Software Process Improvement in small Organizations". ICSP 2009, 342-353

[21] Cruz, R., Morales, M., Morgado, M. Oktaba, H., Ibargüengoitia, G., Pino, F., Piattini, M. "Supporting the Software Process Improvement in Very Small Entities through E-learning: the HEPALE! Project". ENC 09, in press.