8-2010

Core Asset Repository Methodology (CARM) for Software Reuse

Yael Shemesh
RT SW Dept., R&D Missile Division,, yaels@rafael.co.il

Esther Glasser
SW Infrastructure, R&D Missile Division,, ettigl@rafael.co.il

Meira Levy
Ben-Gurion University of the Negev, lmeira@bgu.ac.il

Recommended Citation

Shemesh, Yael; Glasser, Esther; and Levy, Meira, "Core Asset Repository Methodology (CARM) for Software Reuse" (2010). *AMCIS 2010 Proceedings*. 301.
http://aisel.aisnet.org/amcis2010/301

Core Asset Repository Methodology (CARM) for Software Reuse

Yael Shemesh
RT SW Dept., R&D Missile Division, Rafael
POB 2250/39, Haifa, 31021, Israel
yaels@rafael.co.il

Esther Glasser
SW Infrastructure, R&D Missile Division, Rafael
POB 2250/39, Haifa, 31021, Israel
ettigl@rafael.co.il

Meira Levy
Department of Industrial Engineering and Management
and Deutsche Telekom Laboratories at BGU
Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel
lmeira@bgu.ac.il

ABSTRACT

One of the main barriers to overcome when initiating knowledge management (KM) processes within organizations is the structuring of a knowledge repository to disseminate and reuse knowledge throughout the organization. A core asset repository methodology (CARM) is presented in this paper for developing a knowledge repository that encompasses a set of components, which represent abstract design solutions for a family of related problems. The CARM was developed and implemented as a real-time core asset repository (RTCAR) for an actual software development department of a large high-tech organization. The rationale for building the knowledge repository is discussed as well as the technical, managerial and cultural aspects that should be considered when developing it. In addition, the RTCAR findings are presented with examples taken from usage scenarios. The findings show that a knowledge repository constructed according to CARM is conducive to software component reuse, a shortened development cycle and improved software quality.

Keywords

Knowledge management, knowledge repository, software development, software asset.

INTRODUCTION

Software development processes are knowledge intensive, requiring knowledge management (KM) for promoting continuous learning and the exchange of experiences (Feldmann and Althoff, 2001). In this regard, the term “experience factory” is often used in software engineering (SE), when discussing aggregating life cycle experiences, processes and products that are to be packed and stored in a repository for reuse in software development (Basili, Caldiera, and Rombach, 1994). While software organizations have conducted extensive KM research (Aurum, Jeffery, Wohlin, and Handzic, 2003; Chewar and McCrickard, 2005; Bjørnson and Dingsøyr, 2008), there is a lack of research – and knowledge – in how to provide systematic guidelines for creating a knowledge repository for software assets. Consequently, practical implementations accompanied by evidence regarding knowledge repository usage in real settings are also lacking.

The core asset repository methodology (CARM) can be considered as a third generation of knowledge management, where KM is targeted as shared context. The CARM fosters a description and organization of content in such a way that will provide end-users with awareness that content can easily be accessed and applied. This approach stems from the fact that many organization-wide, monolithic digital libraries that were constructed during the first generation of KM were entirely devoid of content (“information junkyards”) and characterized by low usage rates (Dalkir, 2005). The focus now is fostering content reuse in various contexts in the organization. Unless this approach is implemented, knowledge management may be considered a failure (Dalkir, 2005).

We employed the design science research methodology (Havner et al., 2004), in a real setting, where the first two authors work, in developing and implementing CARM. The developed CARM contributes to both KM and SE disciplines by defining a comprehensive methodology that encompasses, on the one hand, the software framework concept (Mili, et al, 2002), and, on the other , the KM aspects that deal with technology, processes and cultural issues.

The rest of this paper is organized as follows: Section 2 presents a theoretical background of KM, specifically with regard to software development processes; Section 3 describes the core asset repository methodology (CARM); Section 4 describes the real- time core asset repository (RTCAR) usage findings; and Section 5 concludes and presents future plans.

KNOWLEDGE MANAGEMENT

Knowledge is considered as the main competitive asset of an organization, enabling the enterprise to be productive and to deliver competitive products and services. There are several knowledge frameworks that describe knowledge-related practices. Holsapple and Singh (2003) describe potential sources of competitive advantages in a firm. They developed the knowledge chain model that identifies five primary and four secondary knowledge-manipulation activities. The primary activities include: knowledge acquisition, knowledge selection, knowledge generation, knowledge internalization and knowledge externalization. The secondary activities include: knowledge leadership, knowledge coordination, knowledge control, and knowledge management (KM).

O'Dell and Grayson (2003) illustrate a comprehensive framework for knowledge transfer that includes the knowledge life-cycle practices (create, identify, collect, organize, share, adapt and use) and the environment – cultural and structural – that is necessary for dynamic and successful KM processes.

One of the cornerstones of KM is improving productivity by effectively sharing and transferring knowledge. However, in practice this may be time-consuming and often impossible (Davenport & Prusak, 1998; Wiig & Jooste, 2003). The organizational knowledge is embedded in people, systems, procedures and products. Knowledge workers are required to improve their work on a daily basis in a process that cumulates in a significant improvement in performance for the entire enterprise (Druker, 1993; Wiig & Jooste, 2003).

KM within Software Development Processes

KM as part of the software development process can be improved by recognizing and managing software development-related knowledge content and structure. The main challenge is to transform implicit knowledge into explicit knowledge, as well as to transfer explicit knowledge from individuals to groups within the organization. Software developers are a critical factor in transforming the implicit knowledge to explicit knowledge since they possess highly valuable knowledge, both implicit and explicit, relating to product and software development, project management and technologies. This knowledge is dynamic and evolves with technology, organizational culture, and the changing needs of the organization's software development practices (Aurum *et al.* 2003).

KM comprises elicitation, packaging and management, and reuse of knowledge in all of its different forms. In particular, KM encompasses software engineering artifacts as code, design, requirements, models, data, standards, and lessons learned. Thus, developing effective ways of managing software knowledge is of interest to software developers. However, it is not well understood how to implement this vision (Aurum *et al.*, 2003).

Research about KM in software development processes has focused on development processes (Aurum *et al.* 2003, Levy and Hazzan, 2009), software architecture (Babar, Dingsøyr, Lago, and van Vliet, 2009) design artifacts (Ben-Chaim, Farchi, and Raz, 2009) to name several areas of interest. However, there are only a few reports on solutions that come from the field and which provide a detailed KM solution. In this paper we illustrate, based on a case study, how a department knowledge repository has been constructed as a dynamic source for software assets that can be reused, not only in the originating software department but in other departments as well. Moreover, as the case study illustrates, since CARM is used to manage a robust knowledge repository it is now being implemented by other units in the organization.

CORE ASSET REPOSITORY METHODOLOGY (CARM)

The core asset repository methodology (CARM) defines a road map for software reuse within a particular domain composed of four main elements: (1) **roles** participating in the processes of the CARM, (2) the **structure** of software assets, (3) the **processes** for identification, packaging, distribution, and usage of software assets, and (4) software development **guidelines**.

There are several basic concepts associated with CARM. First, the *software asset* is regarded as a reusable software artifact that is packed in a specified structure, detailed in Table 2. The integration of a software asset into a software product should be handled as a *black box*, i.e., without any changes by the asset's consumer.

The software assets are packed and tested in a special purpose *Asset Maintenance Area,* and then, handled and maintained within the *Core Asset Repository (CAR),* which represents a *software framework* of a specific domain. A software framework is a set of interacting components that together realize a set of capabilities required by a particular application domain. At a given time, the blueprint of the framework is defined, but only some of the components are realized (Mili, et al, 2002). This *software framework* aggregates software artifacts for further reuse in software projects in that domain. Since the CAR is built as a software framework, changes to one component of the repository may impact other components. Thus, the designers of the framework seek to create a robust architecture.

An asset is not developed, but rather existing software artifacts are identified according to the recurring needs of software projects in the organization. These software artifacts are then packaged into assets of the repository.

The goals of the methodology are as follows:

- Shorter development cycles
- High reliability of the software assets
- Tracking of asset usage, which allows distribution of fixes and improvements
- Simple upgrades of the asset in systems that already utilize the software asset

CARM Roles

Various organizational roles participate in CARM. The following table outlines the participants and their contribution to the process.

Roles	**Contribution**
Product Development Teams	• Identify functional requirements of necessary assets. • Identify software artifacts that are candidates to become assets of the CAR. • Submit change requests and defect reports to asset maintainers.
Asset Maintenance and Management Team	• Adaptation and packaging of available software artifacts • Maintenance and distribution of asset versions
Software R&D Manager	• Overall responsibility for both the product development efforts of the organization, and the creation and maintenance of the asset repository.
CAR Architect	• Responsible for the high-level design of the CAR software framework, for eliciting requirements from the projects and providing solutions within the framework. • Support conceptual integration of the CAR framework into software development projects
Asset Architect	• Support conceptual integration of the specific asset into software development projects.
Information Systems Team	• Maintenance and support of a toolset which implements the CAR methodology and processes.
Asset Control Board (ACB)	A group of software developers, led by the CAR architect, whose purpose is to analyze and decide which existing software artifacts should be packaged as assets for the CAR in response to arising needs for new functionality from software development projects.
Change Control Board (CCB)	Like the ACB, the CCB is also a group of software developers led by the CAR architect. The purposes of the CCB are: to organize

Roles	Contribution
	the requests for changes and the defect reports arising from the use of assets; to analyze the impact of required changes; and to decide on a plan for implementing the changes.
Architecture Forum	This forum, consisting of software team leaders and asset architects, holds recurring meetings where software artifacts are selected for packaging.

Table 1. CARM Roles

The Structure of Software Assets

As previously specified, an asset in the CAR is the result of a packaging process that prepares an existing software artifact for reuse. The asset package contains parts that are provided to asset consumers and parts that are relevant only to asset maintainers. The parts of an asset package are defined in the table below.

Part	Description	Relevant to maintainer	Relevant to consumer
Software source code	The source and header files of the implementation of the asset. This implementation is performed according to the accepted software development processes of the organization. The source is consumed by the asset user as a black box and, for a given version, is identical among all the consumers of the asset.	X	X
Usage examples	Sample application which demonstrates the technique for using the asset in a software project.	X	X
Development environment definitions	Workspace files for various Integrated Development Environments (IDEs) which include, for each IDE, compilation flags, source location paths, and other parameters. This part of the asset may vary among consumers, even though the asset is consumed as a black box.	X	X
Documentation	Each asset package includes documentation of the source code, the asset requirements and design specifications, asset change requests, and a user guide. Each release of an asset version is accompanied by a Software Version Document (SVD) and a summary of fixes and changes included in this version.	X	X
Unit tests	In order to guarantee the reliability and quality of the asset, each asset package includes specific unit tests for its software implementation. The unit tests are used to verify asset updates, and for regression tests. The unit tests are performed by the asset maintainers within the asset maintenance area and not in the project development areas of the asset consumers. Validated unit tests for a software asset implementation are a prerequisite for admission of the asset into the repository.	X	

Table 2. Asset Package Structure

Processes of CARM

The CARM defines processes on two levels, at the repository, top level (see

Figure *1*), and at the individual asset level (specified below). The top level process shown in the figure emphasizes that a potential asset is identified from existing software artifacts, undergoes a process that prepares it for black box reuse, and prepares accompanying items as specified in Table 2. Both these products are stored in a repository and made available for use. The process is iterative, and repeats when an asset is updated in response to the needs of the users.

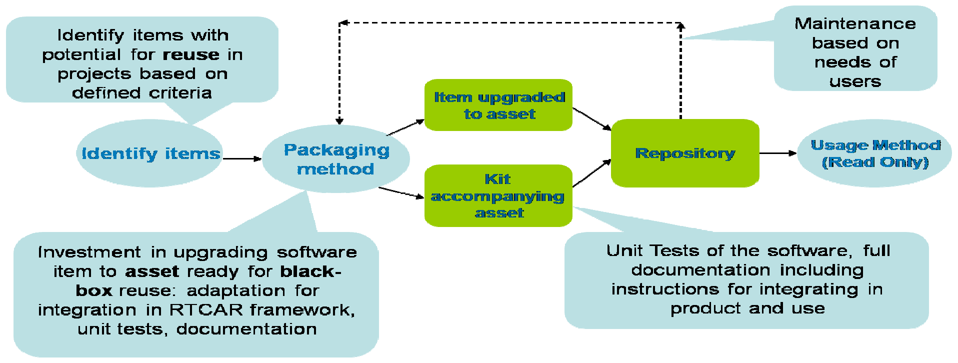
Identify items with potential for **reuse** in projects based on defined criteria
Identify items
Packaging method
Item upgraded to asset
Kit accompanying asset
Repository
Usage Method (Read Only)
Maintenance based on needs of users
Investment in upgrading software item to **asset** ready for **black-box** reuse: adaptation for integration in RTCAR framework, unit tests, documentation
Unit Tests of the software, full documentation including instructions for integrating in product and use

Figure 1. – CAR Repository Top Level Process

Asset Level Process

Identification

The first stage of the CARM process is to identify the need for an asset while observing existing artifacts which may be able to provide the required functionality and become CAR assets. The activities of this stage are specified in the following table.

No.	**Activity**	**Description**
1.	Identify Need for Capability	In the course of software product development, functional components of the application that are not unique to the product are identified. It is probable that this type of component already exists in the CAR, or alternatively, is implemented in another project.
2.	Search for Existing Asset	Before initiating a development effort to implement the functionality identified in the previous step, a search for an available CAR asset is performed in the Asset Knowledge Base.
3.	Search Knowledge Base	The Asset Knowledge Base is composed of asset architects, CAR representatives, and electronic media in a document management system as well as in a configuration management system. All these sources of information are queried for an asset that can implement the required functionality.
4.	Asset Exists	If the CAR contains an asset implementing the required functionality, then proceed to step 3 of the Preparation stage: "Search Knowledge Base for Relevant Asset Version"; otherwise, proceed to step 5.
5.	Capability Exists in Another Project	If a software artifact that provides the required functionality exists in another project, it is examined as a potential asset in step 6.

No.	Activity	Description
6.	Access Control Board	The Asset Control Board convenes to decide whether the software artifact should be packaged as an asset in the CAR. The criteria for the decision are: • The software artifact is fully implemented in the context of the originating project. • The software artifact was developed according to the organization's development policies, including, in particular, documented code and tests at some level. • The software artifact is successfully integrated into at least one application. • The functionality provided by the artifact is general purpose, enough for potential integration in at least three other applications.
7.	Initialize Asset	Kick-off the process of asset packaging as specified in the next section.

Table 3. CARM Identification Activities

Preparation

The second stage of the CARM process specifies the packaging of an asset and its inclusion within CAR. The preparation's activities are defined in the following table.

No.	Activity	Description
1	Asset Preparation	The process of packaging a software artifact into an asset consists of collecting and updating all the parts specified in Table 2.
1.1	Design and Update Code	• Identify existing software artifacts in various products • Elicit requirements for the asset based on the existing artifacts and the needs of the initiating project. • Choose the base of the asset from one of the existing artifacts. • Adapt the chosen artifact that provides a solution for all the requirements, in a manner suitable for the CAR framework, as well as for reuse as a black box. Examples: use of configuration parameters rather than constants; use of object-oriented design methods like inheritance and factories; and parceling the original artifact into several separate components.
1.2	Test	• In order to facilitate black box reuse and increase reliability, comprehensive unit tests are prepared as part of the packaging process. • The asset unit tests may be based on unit tests provided by the originating project if such tests exist.
1.3	Prepare Accompanying Material	As part of the packaging process, documentation, usage examples and Integrated Development Environment (IDE) workspaces are prepared for the asset.
1.4	Release	The packaged asset, or the new asset version, is made available for distribution in the CAR; users are notified as specified in the previous section.
2	Update Knowledge Base	The parts of the asset package are made available in the Knowledge Base.

Table 4. CARM Preparation Activities

Usage

The third and final stage of the asset process includes the use of the asset within a project, the process of reporting about defects and submitting change requests. These activities are defined in the following table.

No.	Activity	Description
1.	Search Knowledge Base for Relevant Asset Version	This step is performed when a project identifies an asset in the CAR which is appropriate to its needs. At this stage, the project needs to pinpoint the relevant asset version.
2.	Consume Asset Version	During this step, the asset is integrated into the software product that is using the asset. This includes changes, if needed, to the software product as documented in order to use the asset interface. The use of the asset version by the project is recorded to facilitate future communication with asset users and to provide for "where-used" reporting. It is possible, that while using the asset, defects will be identified or required changes will arise. In such cases, the users of the asset initiate a Software Problem Report (SPR) which is directed to the CCB.
3.	Change Control Board Activities (CCB)	In order to strive for backward compatibility and to uphold the design of the framework, a controlled change process is performed on asset updates. This change control process is similar to that performed on software products. However, for CAR assets there are additional challenges: • Many stakeholders – all the projects currently using the asset, the CAR architect, and the CAR management team, are all stakeholders of an asset. • At any given time, several versions of an asset may be in use in projects, thus requiring backward compatibility of all changes to the extent possible. • A change in an asset requires re-packaging of the asset. The CCB members classify each SPR according to its level of complexity based on the following parameters: • A change that does not impact interfaces or functionality • A change in the Application Programming Interface (API) of the asset • Documentation changes. • New functionality • Change in functionality

Table 5. CARM Usage Activities

Software Development Guidelines

In order to take advantage of CARM assets, both the project and the asset are developed according to object-oriented analysis and design methods. This facilitates black box reuse by utilizing, for example, inheritance, overloading, and so on. An asset developed for black box reuse (as described in Table 4) has the following advantages, which are not to be found in white box implementations:

- Reliability – an asset that is consumed as a black box has the advantage of having undergone comprehensive tests such as: unit-tests (during the packaging process), and integration/system tests (throughout the asset's consumption).
- Simple to consume updates and backward compatibility retained – known defects in assets are fixed centrally and efforts are made to guarantee backward compatibility. Since assets are consumed with no product development changes, they are easily replaced in the projects.

When a software development project identifies a need for an asset from the repository, the following stages are performed by the project developers before the asset can be integrated into the project:

- At the design level, project developers must understand the relevant part of the software architecture framework. This knowledge is imparted to the project developers by the asset architect.

- Familiarization of the asset interfaces and preparation of the software product implementation for integration with the asset. This stage is supported externally by a CAR representative.

As previously specified, upgrades to asset versions are not imposed on consumers of the assets, but rather, a project using an asset initiates an update to an asset version at a time convenient to the project. When a new version is available in the CAR, the CAR manager notifies asset users of the new release, its version number, changes contained in the release, backward compatibility of the version, and other assets impacted by the change.

Cultural Aspects

The CARM addresses many of the cultural challenges facing reuse of software artifacts in an organization. Attempts to encourage reuse of "generic" components in an organization is often met with skepticism and cynicism on the part of the developers, who perceive reuse processes as potentially beneficial to the organization, but rarely to their own project's goals. Therefore, both the asset users and the asset donors are rewarded with bonus awards and excellence certificates. In addition, since the CARM strategy aims to provide developers with high quality tested software artifacts, the CAR architects are the most experienced developers in the organization. They remain involved with the projects and do not constitute a separate organizational entity. It is critical, that both the developers and CAR architects feel a direct connection and responsibility for the CAR's high standard and success.

CARM BENEFITS WITHIN A REAL-TIME CAR (RTCAR)

This section presents the benefits of CARM implementation within the real-time (RT)/embedded software department of a large high-tech organization. The department includes more than 100 developers and provides software for dozens of projects simultaneously. The motivation for adapting the CARM and building the RTCAR was the rapid increase in the number of products and the demand for a shorter product life-cycle. These goals were to be met without a concurrent increase in personnel. The following sections outline the benefits derived from developing RTCAR according to CARM. The benefits are illustrated with an example of a software asset in RTCAR: the COMM asset for implementing various communication channels between real-time applications and their platforms.

Return on Investment (ROI)

In general, the ROI of an individual asset may be calculated with the following equation:

$$(E_p * P) - E_a$$

where E_p is the product development effort required to provide the required software functionality in a single project; P is the number of projects utilizing the asset; and E_a is the asset packaging effort of an existing software artifact into an asset. In the COMM asset, the development of this functionality for a product (E_p) was 3 man-months; the packaging effort (E_a) was 3 man-months; and currently the asset is used in 7 projects, resulting in a total saving of 18 man-months.

Improvement of Software Quality

For an individual asset

Two factors contribute significantly to the quality and reliability of an individual asset. The first factor is the comprehensive unit tests which accompany the asset. The second is the integration of the asset in many different applications, which are themselves subject to system tests. This is one of the main advantages of black box reuse. In the case of the COMM asset, there is an observable decrease in the number of identified defects with each use of the asset in the projects.

For the software product

As mentioned in the previous section, the CAR is based on a software architecture framework. By integrating an asset from the CAR, a project has in effect adopted the framework of the CAR and thus gained a proven architecture for its design. The RTCAR framework has been successfully adopted by many different types of projects. Some of the projects that adopted the RTCAR framework have successfully undergone major additions, for example, support for a new operating system and low-level drivers, while maintaining their original RTCAR-based architecture. This experience with the RTCAR indicates that the framework is a robust architecture for the type of applications served by the RTCAR.

Improvement of the Software Development Process

Reuse with CARM improves the following aspects of the software development process:

- Rather than arbitrary reuse, reuse is performed systematically as part of the development methodology tailored to the process and framework of the domain.
- The CARM provides several improvements to the software development process. In the RTCAR, the table below indicates the improvements that were exhibited:

Process activity	**RTCAR improvement**
Project kick-off meeting	• Identify existing RTCAR assets which may be utilized in the project • Identify functionality of the project with potential for packaging as assets.
Software Configuration Management Plan (SCMP)	Document method of asset usage and update from within CM tool.
Propagation of knowledge across projects	An architecture forum of software team leaders and asset architects is convened periodically to exchange information.
Tracking of usage of shared assets, to facilitate collaboration and updates.	The methodology specifies that asset usage is tracked, and "where-used" reports may be generated on request.

Table 6. RTCAR Improvements to the Software Process

Improvement of the Support and Maintenance of Software Products

When new functionality is required from an existing asset, the functionality is added according to the asset maintenance process specified above. The requesting project can easily integrate the new asset version after it has undergone unit testing. In addition, all other users of the asset are informed of the improvement available and may choose to acquire the updated version. A project using a CAR asset may reveal a defect in the asset. In this case, a fix is performed within the CAR maintenance area and according to the defined process. Here also, the updated asset may be easily acquired by the project, due to the black box method of reuse. In addition, all other users of the asset may enjoy the updated version of the asset as well. For example, after several uses of the COMM asset, a need to move some of its functionality to a new, separate asset was identified as necessary for portability. This change was performed with minimum impact on the projects using the asset. Projects using COMM after this change gained the benefits of this improvement.

CONCLUSION AND FUTURE GOALS

KM seeks to develop a strategy that helps spread the expertise of individuals or groups across organization in ways that directly affect the bottom line. However, even organizations that have realized the importance of KM are still years away from implementing a solution that filters down to the tactical level (Groff & Jones, 2003). In this paper we present CARM for structuring a software repository. It is based on a solution -- RTCAR -- which was designed, implemented, and maintained for the purpose of reusing generic software artifacts within a particular domain of products. The software artifacts include common mechanisms and types used in the domain's products. Since the artifacts in the repository are documented and tested, they are ready for integration in a product, thus shortening development cycle. In addition, configuration management and change management processes are performed on the assets of the repository which result in a robust and reliable domain architecture that is practically shared within several departments. Future development of the CARM will seek to enhance the search capabilities within RTCAR and to provide tools for collaboration among the RTCAR users.

REFERENCES

1. Aurum, A., Jeffery, R., Wohlin, C. & Handzic, M. (Eds.) (2003). Managing Software Engineering Knowledge. Springer-Verlag, New York.
2. Babar, M. A., Dingsøyr, T., Lago, P., and van Vliet, H. (Eds.) (2009). Software Architecture Knowledge Management Theory and Practice. Springer-Verlag Berlin Heidelberg.
3. Basili, V.R., Caldiera, G., Rombach, H.D. (1994). The experience factory. In: Marciniak, J.J. (Ed.) Encyclopedia of Software Engineering, vol. 1, pp. 469–476. Wiley, New York.
4. Ben-Chaim, Y., Farchi, E. and Raz, O. (2009). An effective method for keeping design artifacts up-to-date. Proceedings of the fourth workshop on Wikis for Software Engineering 2009 in ICSE.

5. Bjørnsson, F.O., Dingsøyr, T. (2008). Knowledge management in software engineering: A systematic review of studied concepts and research methods used. Inform. Software Tech. 50(11), pp. 1055–1168.
6. Chewar, C.M., McCrickard, D.S. (2005). Links for a human-centered science of design: integrated design knowledge environments for a software development process, in: Proceedings of the Hawaii International Conference on SystemSciences, Big Island, HI, United States, pp. 256-266.
7. Chrissis, M.B., Konrad, M., Shrum, S (2003). CMMI: Guidelines for Process Integration and Product Improvement. Addison-Wesley.
8. Dalkir, K. (2005). Knowledge Management in Theory and Practice. Elsevier Butterworth–Heinemann.
9. Davenport, T. H. and Prusak, L. (1998). Working Knowledge, paperback edition, Harvard Business School Press, Boston.
10. Druker, P. 1993. Post-Capitalism Society, UK, Oxford, Butherworth-Heinemann.
11. Feldmann, R.L., Althoff, K.D. (2001). On the status of learning software organizations in the year 2001. In: Althoff, K.D., Feldmann, R.L., Mller, W. (Eds.) Learning Software Organizations Workshop, Lecture Notes in Computer Science, vol. 2176, pp. 2–6. Springer, Kaiserslautern,Germany.
12. Hevner, A. R., March, S. T., Park J., and Ram S. (2004). Design Science in Information Systems Research, *MIS Quarterly* 28 (1), pp. 75-105.
13. Holsapple, C. W. and Singh, M. (2003). The Knowledge Chain Model: Activities for Competitiveness, in Handbook on Knowledge Management, Holsapple, C. W. (Ed.) 2003, 2(43), Springer, Lexington KY, USA, pp. 215-251.
14. Groff, T.R. and Jones, T.P. (2003). Introduction to Knowledge Management: KM in Business, Butterworth–Heinemann, Elsevier Science.
15. Levy, M. and Hazzan, O. (2009). Knowledge Management in Practice: The Case of Agile Software Development. Proceedings of CHASE workshop in ICSE 2009, Vancouver, Canada.
16. Mili, H., Mili A, Yacoub, S, Addy E. (2002). Reuse-Based Software Engineering: Techniques, Organization, and Controls. John Wiley & Sons.
17. O'Dell, C. & Grayson, J. C. (2003). Identifying and Transferring Internal Best Practices, in Handbook on Knowledge Management, Holsapple C. W. (Ed.) 1(31), Springer, Lexington KY, USA, pp. 601-622.
18. Madanmohan, R. (2005). Overview: The Social Life of KM Tools in Madanmohan, R. (Ed.) Knowledge Management Tools and Techniques Practitioners and Experts Evaluate KM Solutions, Elsevier Butterworth–Heinemann, MA, USA, Ch 1.
19. Wiig, K. M. & Jooste A. (2003). Exploiting Knowledge for Productivity Gains, in Handbook on Knowledge Management, Holsapple C. W. (Ed.) 2(45), Springer, KY , pp. 289-308.