AN INTEGRATION-ORIENTED MODEL FOR APPLICATION LIFECYCLE MANAGEMENT

Guenter Pirklbauer, Rudolf Ramler and Rene Zeilinger
Software Competence Center Hagenberg, Softwarepark 21, A-4232 Hagenberg, Austria

Keywords: Application lifecycle management, Software engineering process, Tool integration, Requirements engineering.

Abstract: In the last years a new trend emerged in the software engineering tool market: Application Lifecycle Management (ALM). ALM aims at integrating processes and tools to coordinate development activities in software engineering. However, a common understanding or widely accepted definition of the term ALM has not yet evolved. Thus, companies introducing ALM are usually confronted with a wide range of solutions following different, vendor-specific interpretations. The aim of this paper is to clarify the concept of ALM and to provide guidance on how to develop an ALM strategy for software development organizations. The paper identifies key problem areas typically addressed by ALM and derives a model to relate the solution concepts of ALM to engineering and management activities. The work has been applied in the context of an improvement project conducted at an industrial company. This case shows how the model can be used to systematically develop a tailored, vendor-independent ALM solution.

1 INTRODUCTION

Traditional software development emphasizes distinct project phases and a functionally separated organization with specialized roles. All of these roles have their own separate perspective, are organized along their own processes, use their own tools, and their data and results are locked in their own repositories. The management of software product development faces the challenge to align these different perspectives to the overall business objectives, to enforce consistent processes spanning the different groups, to manage the relationship between the development artifacts produced by the different groups, and to monitor the development progress across the whole lifecycle. Application lifecycle management (ALM) promises to meet these challenges (e.g., (Schwaber et al., 2006)) and over the last few years a large number of ALM solutions have been announced.

The term ALM is quite new. It appeared in the software tool market around 2004 (Azoff, 2007). Its purpose was to emphasize the tool vendors' move towards integrated tool suites covering the whole application lifecycle. However, tool vendors tend to interpret and define ALM differently, often according to their existing or planned marketing strategies. A common understanding or widely accepted definition of ALM has not yet evolved (Kriinen and Vlimki, 2008). Due to the loosely defined scope of ALM, a broad variety of tools has up to now been labeled as ALM solution while, at the same time, it is hard to compare these solutions on an objective basis.

The goal of this paper is to shed some light into the current situation of ALM and to provide guidance on how to develop an ALM strategy for software development organizations.

2 ORGANIZATIONAL CONTEXT AND KEY PROBLEM AREAS

The work presented in this paper has been applied in the context of an improvement project conducted at an international company headquartered in Austria. The company's software products are organized as software product lines, i.e., the standard products are customized according to the needs of individual customers. The products evolve over many years of ongoing development, organized in a sequence of related projects with a floating boundary between development and maintenance.

By analyzing the organizational context, we have identified a number of company-specific problems that we have generalized to the following key problem areas. These key problem areas correspond to

Pirklbauer G., Ramler R. and Zeilinger R. (2009).
AN INTEGRATION-ORIENTED MODEL FOR APPLICATION LIFECYCLE MANAGEMENT.
In *Proceedings of the 11th International Conference on Enterprise Information Systems - Information Systems Analysis and Specification*, pages 399-402
DOI: 10.5220/0002007203990402

issues and obstacles we also observe in many other industrial projects conducted by a variety of different enterprises.

- **Lack of Reuse of Lifecycle Artifacts.**
  When enterprises evolve over time, the reuse of artifacts gets more and more important. Reuse at the level of code has been addressed by object-oriented concepts. The key challenge lies in transforming the reuse paradigm to lifecycle artifacts such as requirements, architecture, design, terms, and definitions.
- **Unclear Rationale of Historical Decisions.**
  Decisions are often based on informal discussions, which are not documented or preserved in any other way. Therefore, decisions can not be traced back to their origins and the rationale for requirements can not be reconstructed.
- **Intransparent Consequences of Changes.**
  Software products are subject to continual improvement and enhancement over many years. Changes and enhancements become part of the day-to-day business and the need for systematic investigation of the impact of changes increases.
- **Imbalance of Activities over the Lifecycle.**
  In many organizations the activities over the lifecycle of software products are imbalanced with an overemphasis on core engineering activities such as coding, testing, building, and deployment. Activities like product management, product portfolio management, asset management are frequently neglected.
- **Heterogeneous Tool-Infrastructure.**
  In the analyzed case, the existing tool infrastructure has evolved over many years. Tool integration has not been seen as an important aspect for the infrastructure. Hence, collaboration of roles using different tools is hampered as the tools isolate activities by intransparent and proprietary tool data stores and repositories.
- **Disrupted Workflows.**
  The limited interoperability of tools also causes frequent disruptions in workflows and processes. As a consequence, we observed a high number of redundant activities as well as data in the analyzed case. This redundancy leads to increased administrative overhead, error-prone manual work, and inconsistencies, which are in turn the source of future errors.
- **Intransparent Status of Artifacts and Work Progress.**
  Due to the heterogeneous tool landscape, artifacts are stored both distributed and redundantly in several tool repositories. Even with careful considerations and policies in place to align workflows it is cumbersome to determine the current status of an artifact or the overall progress of the work.
- **Inability to Reconstruct Historical States.**
  Apart from the fact that the code of the software system is under control of a versioning system, many artifacts are produced and archived in separate tools, that are not able to capture the historic states or use incompatible approaches versioning, baseling, and merging.
- **Missing Integration of Product Management and Project Management.**
  In the analyzed case, the software products evolved over many years. After the initial development phase, the software product will be released and from this point on it is maintained and enhanced in a series of ongoing, parallel projects. This enforces two views. The view on products and the view on projects. Often, however, the management focus is mainly concerned about individual projects, which results in sub-optimal business decisions.

3 INTEGRATION-ORIENTED ALM MODEL

To develop a strategy for introducing ALM in the case company, we developed a conceptual model for ALM, independent from a particular tool vendor's offer. Instead of features and activities, the model emphasizes the solution concepts proposed by ALM and relates these solution concepts to the existing engineering and management activities.

3.1 ALM Solution Concepts

We identified two main goals of ALM by matching the tool vendors' propositions with the key problem areas listed above.

1. Seamless integration of engineering activities at tool and process level across the whole lifecycle of an application
2. Emphasis on management activities to shift the technical perspective of engineering towards the business perspective of software management

These two main goals are refined to a number of solution concepts, which are implemented by tool vendors using central repositories, service-oriented architectures, integrated tool platforms, etc. Following solution concepts have been identified:

- *Traceability.* Traceability is a well-known concept, defined as "the degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor or master-subordinate relationship to one another" (IEEE, 1990).

- *Version Control.* Over the lifecycle of an application multiple versions evolve and require consistent control for managing releases, maintaining defined states and baselines across different artifacts, as well as reverting to these defined states. The concept of version control is well established (see e.g. (Leon, 2000)), whereby research is continuously pushing the boundaries of version control concepts beyond source code artifacts (Estublier et al., 2005).

- *Measurement.* Retrieving or computing information about products, processes and resources as well as their relationships is the basis for establishing transparency, objective evaluation and planning of future activities. The role of measurement has been firmly grounded into the management of software projects by DeMarco (DeMarco, 1986) and has been expanded to the entire application lifecycle.

- *Workflow Support.* By execution, workflows bring together a sequence of operations, resources, roles, and information flows to achieve a result of value. Traditionally, workflows have been focused on single activities and tools, for example, issue tracking or requirements management. Current approaches, such as the Application Lifecycle Framework (ALF) (Eclipse, 2008), provide a interoperability platform for tools to establish a workflow across the entire lifecycle of an application.

- *Collaboration.* Software development is a team endeavor and, thus, concepts and tools for collaboration, e.g. Wikis have found their way into software development. Computer-support for collaboration and cooperation is yet again another long-established field (see e.g. (Grudin, 1994)).

- *Shared Services.* In addition to the above concepts we identified a number of basic services which are relevant for every activity and tool applied in the course of ALM. A typical example is managing users and access rights, which are often implemented as a central service shared among the different activities and tools.

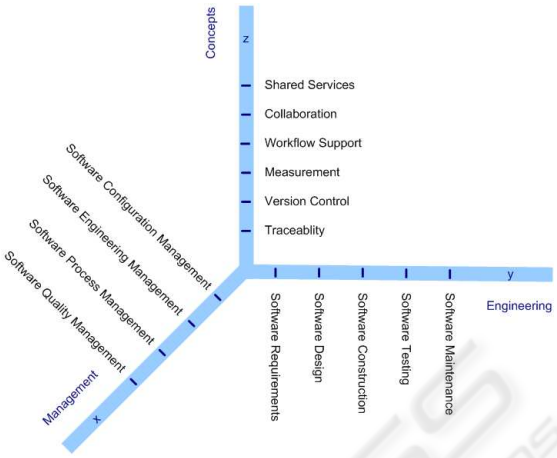

Figure 1: Integration-oriented ALM Model

3.2 Dimensions of Integration

The identified solution concepts have a strong focus on integration in two directions. First, they provide profound support for the integration of engineering activities over the whole application lifecycle at tool level as well as process level. Thereby they address the first goal of ALM. Second, the same concepts also enable the integration of engineering activities with management activities. In that way they establish the link between the technical and business perspective along the application lifecycle, the second goal of ALM.

Figure 1 depicts these three dimensions showing the engineering activities (software requirements, software design, software construction, software testing, and software maintenance according to (Abran et al., 2004)) on the x-axis and the management activities (software configuration management, software engineering management, software process management, software quality management according to (Abran et al., 2004)) on the y-axis. The solution concepts of ALM (traceability, version control, measurement, workflow support, collaboration, and shared services according to Section 3.1) are shown as third dimension depicted as z-axis to clearly distinguish them from engineering and management activities. This perception helps to clarify the contribution of ALM as the thread that ties the different activities of engineering and management together.

4 IMPLEMENTATION

The ALM model has been used to find an ALM strategy for the analyzed company. Therefore, the prob-

lems were mapped to combinations of engineering and management activites. The concepts were priorized according to the estimated value for supporting the found combinations. As a consequence, the ALM solution concepts *traceability* and baselining (part of *version control*) have been implemented in the company. We have made changes on two different levels of the company's software engineering structure. Firstly on the process level and secondly on the tool-infrastructure level.

On the process level the company had neglected some activities of requirements engineering, since the time pressure in the projects did not allow a detailed documentation of requirements and related decisions. The implemented solution is based on an information model representing new artifacts. These documents enforce the documentation of all relevant information and consistently describe the state of the entire product, which previously had to be digested from a long list of descriptions of incremental changes made over time.

On tool-infrastructure a requirements management tool had been introduced as central integration point for the new artifacts. Hence, the information previously contained in different repositories is now hosted under a single roof, avoiding inconsistencies as well as unnecessary overhead for maintaining baselines and traceability.

5 SUMMARY AND OUTLOOK

In this paper we have identified key problem areas of enterprises in the software engineering business. Based on these key problem areas we have elaborated a model of ALM, which integrates engineering activities and management activities. This model can be operationalized to systematically develop a company-specific ALM strategy and to trigger process improvement projects. Furthermore, the model fosters a common view for the evaluation of ALM solutions.

An example about the application of the model for introducing ALM in a software development company has been provided. Preliminary results confirm the applicability of the model as a guidance for identifying and prioritizing problem areas as well as planning a tailored ALM solution. Our plans for future include the repeated application of the model for other companies in different domains and to establish a vendor-independent improvement approach leveraging ALM for small and medium enterprises.

REFERENCES

Abran, A., Moore, J. W., Bourque, P., Dupuis, R., and Tripp, L. L. (2004). *Guide to the Software Engineering Body of Knowledge (SWEBOK)*. IEEE.

Azoff, M. (2007). The future of application lifecycle management. *OpinionWire Articles*. Butler Group.

DeMarco, T. (1986). *Controlling Software Projects: Management, Measurement, and Estimates*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Eclipse (2008). Eclipse Application Lifecycle Framework (ALF) Project. http://www.eclipse.org/alf/ (November 2008).

Estublier, J., Leblang, D., van der Hoek, A., Conradi, R., Clemm, G., Tichy, W., and Wiborg-Weber, D. (2005). Impact of software engineering research on the practice of software configuration management. *ACM Trans. Softw. Eng. Methodol.*, 14(4):383–430.

Grudin, J. (1994). Computer-supported cooperative work: history and focus. *Computer*, 27(5):19–26.

IEEE (1990). IEEE standard glossary of software engineering terminology. Technical report, IEEE.

Kriinen, J. and Vlimki, A. (2008). Impact of application lifecycle management - a case study. In *Enterprise Interoperability III - New Challenges and Industrial Approaches*. Springer London.

Leon, A. (2000). *A guide to software configuration management*. Artech House, Inc., Norwood, MA, USA.

Schwaber, C., Rymer, J. R., and Stone, J. (2006). The changing face of application life-cycle management: Tomorrow's alm platforms will deliver on the promise of today's ALM suites. Technical report, Forrester.