Towards an industrial ALM (Application Lifecycle) Tool Integration

Georgi A. Markov
Olga R. Druzhinina

Thesis submitted for completion of Master of Science (120 credits)
Main field of study: Computer Science
Specialization: Informatics

June 2011
This thesis is submitted to the School of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science (120 credits) in Computer Science with specialization in Informatics. The thesis is equivalent to 20 weeks of full time studies (30 credits).

Contact Information:
Author(s): Georgi A. Markov
E-mail: markov.georgi@gmail.com
Olga R. Druzhinina
E-mail: krysaseraya@gmail.com

University advisor(s): Prof. Stig C Holmberg, PhD
                      Prof. Anita Håkansson, PhD

                       Prof. Sara Eriksén, PhD

Department of Informatics, Mid-Sweden University
School of Computing, Blekinge Institute of Technology

School of Computing
Blekinge Institute of Technology
SE-371 41 Karlskrona
Sweden

Internet: www.bth.se/com
Phone: +46 455 38 50 00
Fax: + 46 455 38 50 57
ABSTRACT

Context. Application Lifecycle Management is an important collection of various tools, targeted at assisting the business processes at every stage of life of a software application. It is concerned with coordinating and managing of the lifecycle activities. More and more organizations are heavily relying on the ALM tools nowadays. ALM fulfills the managing of all tasks of the software producing. ALM saves the economic resources and helps prevent mistakes.

Objectives. In this study we research ALM first as a business strategy, its roots, its advantages and disadvantages. As well as making a theoretical research, we seek out the setbacks of the practical side of ALM in order to find the areas needed for improvement, by investigating the ALM’s existing tools, interviewing ALM’s users, and using gained statistical data to derive practical facts about ALM. The goal of the thesis is to find out how ALM can be improved.

Methods. The vast amount of sources, including articles, books, and journals is used to support our arguments and conclusions. A long online survey was carried with many participants to understand the practical side of ALM, as well as gain access to information about ALM’s setbacks.

Results. The problems of ALM are recognized and named. Proposals are made in respect of how to deal with these problems. A list of what options need to be studied in the future, in order to improve the overall ALM design, and make it a powerful tool, is presented. A detailed plan for further research regarding this highly important issue is presented.

Conclusions. We conclude that there are several ALM tool options on the market with competition high and variable offers. But in order for ALM to become an industrial solution, the requests of the stakeholders need to be embodied in the ALM structure. We conclude that further research is needed in order to assess the proposals, and test the design ideas that are presented in the last chapter of this thesis.

Keywords: Application Lifecycle, Software Development Lifecycle, Application Lifecycle Management, Tool Integration.
EXECUTIVE SUMMARY

This thesis concerns the authors and stakeholders of ALM. With the technological achievements taking place every day, and competition being very high, it is important to get the software product ready for release on time and of good quality. The current tools for managing of the application’s lifecycle are incoherent in terms of tool integration. In this thesis we describe the setbacks of ALM, according to interviewed stakeholders and authors of journals, articles and books. We conclude our thesis with the several proposed design options that may succeed in making ALM strong and consistent in the future. The proposed design mainly concerns the integration problems, and provides several potential solutions, aiming at easy to use and integrate tools. Further research areas are explained, and stated.
# CONTENTS

TOWARDS AN INDUSTRIAL ALM (APPLICATION LIFECYCLE) TOOL INTEGRATION ................................................................................................................................. I

ABSTRACT ........................................................................................................................................................................ III

EXECUTIVE SUMMARY .................................................................................................................................................. IV

CONTENTS ..................................................................................................................................................................... V

1 INTRODUCTION ......................................................................................................................................................... 1

2 RESEARCH PROCESS ................................................................................................................................................. 3
  2.1 The research environment ........................................................................................................................................ 3
  2.2 The research methods and the research process .................................................................................................. 4

3 THE ALM BASICS ....................................................................................................................................................... 6
  3.1 The Application Life Cycle ........................................................................................................................................ 7
  3.1.1 The Software Development Life Cycle .................................................................................................................. 9
  3.1.2 IT Maintenance and Operations ............................................................................................................................. 14
  3.1.3 Application Portfolio Management ..................................................................................................................... 15
  3.2 ALM Tool Integration and Coping with Complexity in Software Development ....................................................... 17
  3.3 Basic Concepts and Techniques of Tool Integration ............................................................................................... 18
  3.4 The ALM Approach to Tool Integration .................................................................................................................. 20

4 APPLICATION LIFECYCLE MANAGEMENT TODAY ......................................................................................... 22
  4.1 What is ALM today and how does it relate to lifecycle approaches from other fields .............................................. 22
  4.2 What are the advantages with the current form of ALM ....................................................................................... 26
  4.3 What are the disadvantages with the current form of ALM .................................................................................. 27
  4.4 Why do we need a further research in the area of ALM ......................................................................................... 28

5 ALM 2.0 – DESIGN FOR THE FUTURE ......................................................................................................................... 32
  5.1 Identification of relevant ALM stakeholder (stakeholder analysis) ......................................................................... 32
  5.2 Elicitation of future ALM requirements through stakeholder interviews and questionnaires .................................... 34
    5.2.1 Interview Questions ............................................................................................................................................. 34
    5.2.2 Results of the elicitation ....................................................................................................................................... 38
  5.3 Identification of improvement areas based on a delta analysis of current ALM solutions and the elicited requirements for a future ALM approach .............................................................................. 41
    5.3.1 Microsoft Visual Studio with Team Foundation Server .......................................................................................... 41
    5.3.2 SAP Solution Manager ............................................................................................................................................ 45
    5.3.3 BMC Remedy IT Service Management Suite ........................................................................................................ 47
    5.3.4 Borland StarTeam & the Open ALM concept ........................................................................................................... 48
    5.3.5 Rational Collaborative ALM (C/ALM) & IBM‘ Jazz Integration Architecture ..................................................... 51
    5.3.6 HP ALM 11 .............................................................................................................................................................. 53
    5.3.7 Summary ................................................................................................................................................................. 55
  5.4 Design concepts and ideas ............................................................................................................................................ 56
  5.5 Identified grand challenges and further research needs ............................................................................................ 58

6 VALIDATION OF RESEARCH RESULTS ..................................................................................................................... 60
  6.1 Validation of requirements elicitation results ........................................................................................................... 60
  6.2 Validation of tool evaluation results ........................................................................................................................ 61
  6.3 Validation of design ideas ............................................................................................................................................ 62

7 OUTLOOK ..................................................................................................................................................................... 63
7.1 SUMMARY .............................................................................................................. 63
7.2 VERIFICATION OF PROJECT GOALS ................................................................. 64
7.3 NEXT STEPS ......................................................................................................... 65

REFERENCES .............................................................................................................. 66

APPENDICES ............................................................................................................. 72
APPENDIX A ............................................................................................................. 72
1 INTRODUCTION

In recent years, software has become more and more complex. In order to cope with this increased complexity, multiple levels of formal and logical abstractions have been introduced, each producing different set of artifacts representing a different view on the final product. Examples of such abstraction levels are requirements management, architecture and design, implementation, test management, build management, release management, issue management, etc. All of these different stages in the life of every software product are each supported by different development or product information management systems like requirements management databases, development tools, test tools, modeling tools, variability management tools, issue management tools, configuration management tools and others. Each of these tools targets only its very specific development lifecycle phase, however in order to successfully coordinate and control the software development activities, the artifacts and objects produced by these tools need to be interlinked and traced among each other for the purpose of process automation, reporting, impact analysis, regression testing, etc. and to help administer a cohesive and comprehensive software development process (Kääriäinen et al., 2009). Unfortunately it is often very difficult to amalgamate all those different tools and to make them to exchange information, while still preserving the semantic and the consistency of the data, which turns the challenge of finding a way to integrate all these different aspects of software development toward delivering high quality, long lasting and business critical products into the holy grail of modern software development.

This is where Application Lifecycle Management (ALM) comes into play. ALM provides an ecosystem of integrated tools, processes and domain technologies aimed at increasing the consistency, predictability and measurability of the software development process and beyond by coordinating, managing and keeping in sync the different involved activities as well as providing unification and automation for each of the major participating roles and stakeholders (Carrillo & McKorkle, 2008). In such a way by integrating development, collaboration, communication and knowledge management tasks and centralizing management of users, projects and processes, ALM can be the answer for the challenges of distributed software development, as it represents a giant leap towards controlled, manageable and auditable processes across all the different software development stages and beyond, which itself is an important step towards a full process automation and efficiency.

Most of the currently available unified ALM solutions are either based on basic version control and other 'low level' point-to-point integrations, or advocate the adoption of a new and expensive all-in-one solution from a single vendor. The problem with these current solutions is that the first does not go far enough to really provide the previously described benefits of applying an ALM solution, while the second one is often associated in high costs in tools, infrastructure and personnel. This thesis investigates a third option, which would allow increase in the value of software investments for organizations at flat or only slightly increased spending by providing deep semantic integration between existing tools and processes, organizations have already committed to, both financially (in terms of license-fees, etc.) and knowledge-wise (in terms of workforce trainings to master these tools).

Although the main idea behind ALM, namely that a common infrastructure should exist, which helps to centralize, organize and align all software-related processes and tools and align their capabilities, is not new, it has only established itself as a autonomous discipline recently. In the past this idea of harmonizing the many software processes and tools has been pursed through tool integrations, which can be seen as the origin of ALM.
In the past decade, however, the initial requirements for tool integration have changed significantly, as a result of an increased concern on long-term dependence on a single tool supplier, due to incompatibility issues and the high costs associated with switching tools. This has led to an increased effort in the industry on creating and establishing standards, that can be endorsed by tool suppliers (T. Wasserman, 2004). This new situation makes the authors believe, that it is on the time to reexamine the initial requirements on tool integration from a technical perspective, but also to provide a valuable business and defined scope case for tool vendors.

The aim of this thesis project is to elicit the requirement for an integrated ALM solution, based on a series of consistent and pragmatic steps, with the final goal to make an initial contribution towards an tool or framework, which will enable deep semantic integration between different tools (S. L. Wright, 1994) while still considering the business interests of organisations. The thesis shall serve as the first phase in a development project aimed at defining a framework or setting up a research agenda, that shall allow tools to provide consistent quality in the software engineering ecosystem of an enterprise by meeting the critical needs for improved traceability, visibility and automation and by providing a context-aware way to collaborate, communicate, manage and exchange knowledge and information. It shall consider and if appropriate build upon already available solutions. In order to achieve this aim the project team will try to address the following sub-goals:

- Understand, who are the main stakeholders of an ALM solution, in order to be able to match the creative work with the authors aspirations.
- Understand the requirements posed by industry on a integrated ALM solution, in order to ensure that the final solution will meet their needs.
- Evaluate currently available ALM solutions against the requirements, so that available reuse potentials can be exploited.
- Identify the problems with applying currently available ALM solutions and tool integration frameworks in industrial settings, in order to be able to create solutions for negating the problems.
- Define the next steps, aimed at addressing the improvement suggestions.
2 RESEARCH PROCESS

The thesis will present a requirements-driven approach for analyzing the current state of the art of ALM, aimed at providing a deep insight into the strength, weaknesses, opportunities and threats of such a solution. It will present the requirements posed by the industry on the leading ALM products and derive a list of concrete problems, not addressed by them. A real-life interview/questionnaire, from which concrete requirements and current pain points of ALM will be derived, will be carried out and presented in the Appendix. The final deliverables of the project are:

- Model of high-level requirements for an industrial ALM solution from business, user and system perspective (see Appendix E).
- List of stakeholders, derived from a stakeholder analysis (see Appendix B).
- A list of problems not addressed by current ALM solutions as a result of a delta analysis of the features provided by available ALM solutions and the elicited requirements (see Chapter 4.5).
- Draft research agenda focused at solving the identified problems, and suggesting improvements to the current state of ALM (see Appendix F).

The reminder of this chapter presents the design of this research. In it describes the research environment (Section 2.1), the adopted research methods and how they were integrated into the whole research process (Section 2.2).

2.1 The research environment

This study is motivated by problems encountered in real-life projects, conducted by several teams in a business unit, belonging to a global electrical and electronic engineering company. The company has its headquarter in Munich, Germany and a customer base in more then 200 countries around the world, which is served by several hundred thousand employees working to provide innovative technologies and comprehensive know-how.

The business unit, where these problems surfaced, is a research and technology oriented unit within the larger company. The objective of the business unit is to be an innovation partner and to provide in-house consulting services to the other operating business units. Its top priority is to secure the technological future of the company considering prevailing mega trends. Special focus is put on innovations that have the potential to last and alter the fields of interest to entire company. Therefore it supports the other business units in putting their innovation strategy into practice by providing expertise in the form of consulting and innovation support in strategically important technology fields, development of future scenarios and safeguarding business through patent law. In such consulting projects the employees of the business unit often encountered the problematic of a missing integration between the different tools and process control mechanism, utilized in the different phases of the application life cycle, which the authors try to address in this work.

Finally the particular team and project we are primarily concentrating on (as an information source for our interviews and surveys, although information was obtained also from other sources) in our case study is, currently piloting a new concept for automated test generation from UML-models and is developing a tool to support this concept. It is comprised of several developers and consultants in Munich, Erlangen, Princeton and St. Petersburg, all closely integrated and involved in intensive global software development, effectively creating a virtual team across multiple sites. The software product developed is supplied to internal, as well as external customers in the US, Europe, Russia. Additionally the team provides consulting, training and coaching services on concepts and techniques necessary for the effective tool usage (including
UML-modeling courses, test management concepts and architectural concepts, necessary for developing tool extensions). Thus a close collaboration between all four locations involves all SDLC activities, but also operation and service management and customer relationship management and portfolio maintenance management activities.

2.2 The research methods and the research process

As already mentioned in Chapter 1, this work aims to address the issues with tool integration towards constructing an integrated, and industrially feasible, in terms of economic and financial considerations (licensing, workforce training, loss of know how), approach for application lifecycle management. This involved, in the first place, the identification of the stakeholders of such a ALM solution, their respective needs, goals, and requirements. Based on these consolidated findings on what an optimal, according to the involved stakeholders, ALM system is expected to offer in terms of functionality, quality, price, etc., the authors analyzed, in a second step, already existing on the market major ALM systems and platforms with the goal to identify the gap between an optimal and the currently existing solutions. This information was finally used to determine future design objectives and next steps.

This implied the use of stakeholder analysis, a technique used to assist in identifying the range of stakeholders, who are likely to use a system, or to be directly or indirectly influenced by its use. As part of the much wider design philosophy called socio-technical systems theory, stakeholder analysis is based around the fact, that when technical systems are developed and implemented they also affect the people and the organizations that adopt them. Therefore the technique is used to identify the stakeholders, whose views should be consulted, but was also, in our particular case, helpful to determine the stakeholders roles in the system, and the key groups who need to be involved in the following requirements elicitation activities (Robin-Prévallée et al., 1998). The stakeholder analysis was conducted at the organization’s side in the form of a few consecutive informal expert interviews and resulted in an initial list of all parties (in terms of project an organizational roles), who have a ‘stake’ in the decision for an ALM system.

Using surveys and informal, semi-structured interviews to compile a collection of stakeholder requests, which were later on analyzed to derive concrete use cases and non-functional requirements. These interviews and surveys tried to involve represented of all previously identifies key stakeholder groups (not only from the people involved in the previously described project, but also external users, customers and affiliated organizations) and based on the information derived from these, we performed an Use Cases analysis, which resulted in an UML Use-Case Model, which defines how the system is to be used. The questionnaires and surveys consisted of predefined set of questions derived from an extensive initial literature review (both open and closed questions to gather both qualitative and quantitative data) on different topics ranging from definitions and scope of ALM to specific features and functionalities, necessary to meaningfully support particular roles involved in and/or activities, part of the application life-cycle. These questions were presented both verbally, in the case of semi-structured interview, and administered as a web-based interactive tool that was available on the Internet, so that users could complete them either on their own, or in consultation with colleagues. An important side note on this is that, the surveys were the primary information collection technique, due to the fact that they can easier reach many more participants and because the use of “standardized” questions allows the findings from a number of respondents to be much easier summarized, where interview were only used in a few cases (e.g. higher management), where the participants did not have the time or will to answer all questions.

The in such way collected stakeholder requests, needs, goals and problems were used for the derivation of concrete use cases and non-functional requirements and to obtain a granular, component-based of the end-system, ensuring that all participants’
needs are served. Additionally the use of a use case driven requirements analysis would be helpful not only for the current goal of specifying the systems functionality, but also for the further activities directed towards finding out the optimal design for a future ALM system. For the purposes of our work, we abstained from creating full-featured use cases descriptions, including all the different execution scenarios of the particular user-system interaction, pre-/post-condition, etc. as described in (Cockburn, 2001). Instead we created use case models, structured according to the different application life cycle phases and describing on a very abstract and implementation-neutral level features as defined by (Berenbach, 2004), as well as interactions and responsibilities of a future ALM system with external agents, or actors and can be associated with the particular goal one specific actor is pursuing. The use cases consisted of a use case name and a use case description. Non-functional aspects of the ALM systems, which could not be expressed in a meaningful way using use cases, were expressed using SysML Requirements diagrams. The decision to use UML & SysML, rather than simple textual description, although according to (Berenbach, 2004) the latter are easier to develop, was due to the fact that Use Cases models allow to visualize relationships, between use cases, and are less ambiguous and thus allow for automated further analysis. Additionally such diagrams can be easily used to communicate with stakeholders.

In a next step we performed an evaluation and analysis of existing ALM solutions aimed at comparing the elicited requirements against the currently available ALM solution and in such a way to identify problems and gaps, which need further investigation. As part of this tool analysis each tool was briefly described (based on information provided from the tool vendor and experience reports of tool users), explained how it implements the application life-cycle, and then analyzed and criticized by the author in regards to the elicited stakeholder requirements, needs and wishes.

Based on this critique the last step involved several brainstorming session within the thesis team and with representants of the customer’s organization in order to generate an initial list of ideas and design improvements. Brainstorming is a technique designed to help creative thinking in initial system development, where a large number of ideas is generates, and later on evaluated, as part of which many are discarded. In this process however often novel ideas, which are worth following up, emerge, as we hope was the case in our project (Robin-Prévallée et al., 1998).

Finally all research findings were validated in several walkthrough, review and focus group sessions.

The following graphic summarizes how the different methods were integrated into the research process:
3 THE ALM BASICS

While, as already mentioned, the area of ALM is relatively new, and there are not so many peer-reviewed publications on the subject, the topic of tool integration has been subject of active study since the 1980s. Two particular texts focusing on this topic stand out as very influential and inspiring and serve as a basis for the work of several other scholars focused in this area (Wicks, 2005; Wicks & Dewar, 2007): the reference model for CASEE (CASE environments) by Earl in 1989 - 1990 (Earl, 1989) and Wasserman’s 5-layer framework introduced in (A. Wasserman, 1989).

Roughly summarized Earl’s reference model advocates the division of the environment into a toolset on the one side, which provides the specialized facilities required by a particular user, and a framework on the other side, which provides the common facilities required by the entire environment. The common facilities are subdivided into five groups and encompass front-end services and back-end data management services, between which the tools can be slotted as shown on Figure 1 (H. Barker, 1995). While this model provides a useful starting point for identifying logical integration aspects of a holistic architecture towards an integrated ALM tool chain, it, on the downside, does not envision the use of tools outside the framework (Wicks, 2005), which in the authors opinion is a rather severe restriction for a development process, which needs to be able to quickly adjust to today’s highly flexible business environment demands. Another problem, as pointed out in (Chen & Norman, 1992), is the fact that the framework deals with the tool integration subject from a very generic, and solely technical, perspective without considering the organizational context, like used methodology and/or process.

Wasserman proposes a method based on five different dimensions, based on which the level of tool integration in an organization can be measured (A. Wasserman, 1989):

- Platform, referring to the level of physical integration level.
- Presentation, referring to the level of user interface integration.
- Data, referring to the level of integration of the original database schemata of the different tools and the new shared database model.
- Control, referring to the level of interoperability between the tools.

![CASEE Reference Model](Earl, 1989).
• Process, referring to the level of integration between the methodologies supported by the tools.

The model again show the purely technological viewpoint taken in the academe towards solving the problem of tool integration.

According to Wicks, all subsequent works in this area are built on these two seminal contributions toward defining other more sophisticated integration environments without reexamining the initial tool integration requirements. This has led to the emergence of many integration approaches focuses solely on the technical side of the problem and ignoring issues like business impact and organizational circumstances. The entire issue however is “more subtle, then just simple tool to tool communication” (Rossberg, 2008) and the integration strategies “must be both flexible and adaptable to suit differing users” (Maalej, 2009), which in the author’s opinion is one of the major reasons, why after so many years of research the problem has not yet been satisfactorily solved.

In order to address this gap, the authors will firstly conduct a thorough literature review of scientific articles and books to understand the domain backgrounds and to seek out the theoretical setbacks that ALM poses.

A second step will concentrate on carrying out a stakeholder analysis in order to identify the major stakeholders of an ALM solution. A scientific study will be conducted to identify their requirements and to better understand their problems with ALM.

In order to identify what the top ALM products offer and lack, a scientific evaluation will be performed on the most popular ALM tools against criteria derived directly from the elicited stakeholder requests and requirements.

At this point the authors will brainstorm, and try to come forward with proposals of our own on how to best address the identified problems. The interviews/questionnaires will also address the question of how ALM may be improved, and the authors’ proposal shall be lifted in it. Once the solution is formulated and stated, a graphical representation will be used for a better comprehension reason, and as a descriptive summary.

The thesis will conclude with a discussion of what the ALM may offer in the future, based on the objectives and goals of vendors, the new proposal of the authors, the literature reviews, and personal experience of the interviewees.

According to (Rossberg, 2008) when talking about ALM, we need too take into account that there are different perspectives to look at ALM, and the chosen perspective determines the scope of the discussion. Therefore we need to differentiate between these different views and the unified ALM, which is in the focus of this work and represents the ultimate goal for an organization as it comprises all other views and aligns them with the organizational business goals. In this context it covers the entire history of a software application, from the initial vision of the application to its removal from an organization’s systems. The following three perspectives represent the most common separations of the unified ALM in most organizations:

• From a software development lifecycle point of view, ALM focuses mainly on keeping the development activities in sync.
• From a service management and operations point of view, ALM focuses on the activities ensuring that once build the application meets the service level that has been defined to it.
• From an application portfolio management point of view, ALM focuses on all activities involved with managing the application as part of a portfolio of products.

These three different perspectives on ALM will be briefly discussed in the following section.

3.1 The Application Life Cycle
The application, or in other words the product of the software development process, is never developed in one part. Instead the making of such a complex product consists of several stages. According to (Schechner, 2010) the application life cycle can be roughly divided into the following three major phases: everything starts with the initial idea to create a software that provided certain functionality to its users. This phase is referred to as the **conceptualization phase** and involves (besides conceiving the initial idea for the software product) understanding the current state of the business with respect to specific domain in which the application will be deployed, setting up objectives, project scope, project plan, etc. Once all this has been done and the initial idea is put into action, the software or the application is considered in **development**, which is the second major phase in the application life cycle. The development phase involved all activities necessary to produce a running final product, which meets the customer’s and end with the software’s or application’s operationalization and rollout, also called **deployment or IT maintenance and operation**, which is concerned with the day-to-day management of the IT infrastructure and processes. Once the application has been deployed, however the application life cycle does not come to an end. With most applications, issues and bugs arise after deployment and new ideas for updates and further development are generated during the application’s day-to-day use. This leads to the software going back to development, to start over again. The following graphic shows the different stages in the application life cycle.

![Application Life Cycle Diagram](image)

**Figure2: The application life cycle**

Each of the previously described major phases of application lifecycle has it’s own lifecycle and sub-phases.

The process of managing this cycle and other aspects, important for the successful software development like governance and application portfolio management is called Application Lifecycle Management, or ALM. It brings together business management (in terms of time, money, function, market and practicality) and software development and makes them to work together to form something greater that the sum of its parts. In such a way it can ensure that technical decisions are controlled by business events and allows technical stuff to focus on business requirements and maintain consistent quality. These different aspects of ALM are described in the reminder of this section.
3.1.1 The Software Development Life Cycle

The Software Development Life Cycle (SDLC) is a multi-progression process including all activities involved in the software development from requirements through design, development, testing, maintenance and retirement, which is used to help ensuring the successful software development, software operation and retirement (OnPoint-Corporation, 2010).

The picture below shows the different steps, involved in the SDCL life-cycle, without putting them in the context of a specific process model.

Systems Development Life Cycle (SDLC)
Life-Cycle Phases

There are several existing SDLC methodologies, developed to guide the processes involved, including the waterfall model, spiral model, V-Model, incremental, agile, rapid prototyping, etc. As different organizations and even different process have different preferences about the underlying SDCL methodology, ALM needs to offer support for all major forms and/or to be flexible enough to be able to quickly and easily adapt itself to the SDCL methodology applied by any particular organization. The reminder of this section presents the most popular SDLC methodologies.

- Waterfall model
  In his book Tayntor explains, that “because the phases were viewed as sequential steps, with the output from one phase becoming the input to the next, a traditional SDLC was often called a waterfall” (Tayntor, 2003). The idea behind the “waterfall” was to take each phase step by step, but never return to it, after it is finished till the end of the process, as shown on the following graphic.

---

Figure 3: The SDLC steps (Wikipedia, 2011a)
The waterfall model still “continues to provide benefits for many organizations” (Tayntor, 2003). The main advantage of this strategy is that it is simplistic and conceptually straightforward, and therefore the employer does not have to hire experienced workers. The other advantage is that the waterfall model enforces a very disciplined approach and is very documentation driven, and documentation is produced at every step. In such a way “the methodology promotes consistency among projects, which can reduce the cost of ongoing support and allow staff to be transferred from one project to another” (Tayntor, 2003).

The downsides of the waterfall are that it only works in organizations and/or projects where the technical team has a lot of domain knowledge and knows exactly what has to be done, where the requirements are well known from the beginning and do not change to often. In very rapid projects with changing requirements the waterfall model produces the worst results, as it's sequential nature does not allow any flexibility to react on change.

As a result, later the versions of waterfall started to vary from company to company. So called, improved versions of waterfall started to appear, as companies started to adapt this methodology, in order to allow for more flexibility. The one below is one such example:
• **Prototyping model**

The main goal of the prototyping model is to address the limitations of the waterfall model by creating a prototype early in the development process in order to better understand, validate and verify the customer requirements, which minimizes the necessity for changes in the requirements. The prototype itself goes through the phases design, implementation and testing, however these activities are performed with minimal formality. Once the initial prototype is ready, the customer can evaluate it and depending on the outcome of this evaluation the prototype is either refined towards the final product or redesigned to include some new or misunderstood requirements. The following graphic illustrates this process:

![Prototyping model diagram](image)

Figure 6: Prototyping model.

• **Spiral model**

The Spiral Model is another form of SDLC, which combines elements of the controlled and systematic waterfall model and the iterative prototyping model, and in such a way provides advantages of both, the top-down and bottom-up development concepts (Wikipedia, 2011b). The main idea behind the spiral model is to develop
software in a series of incremental releases. The model itself is divided in the following task regions (Al-Masa'fah & Meligy, 2008):

- Objective setting, where the objective for the phase are identified.
- Risk assessment and reduction, where risks are assessed and mitigation procedures put in place.
- Development & validation, where the development model for the software is chosen.
- Planning, where the current project status is reviewed and the next phase of the spiral model is planned.

The spiral model begins at the center and moves in a clockwise direction, and each traversal of the spiral should result in a deliverable. The following graphic illustrates the model:

![Spiral Model](image)

**Figure 7: Spiral Model (Beidler, 2007)**

According to Tayntor, “because the functionality of each iteration is limited, IT can deliver the system quickly”, and another advantage is that “the system’s functionality can be altered as business requirements evolve rather than waiting until the full system is completed” (Tayntor, 2003). Another important advantage is the use of prototyping or storyboarding techniques, which minimized the risk of building the wrong product (by receiving customer feedback) already very early in the project (Al-Masa'fah & Meligy, 2008).

Amongst the disadvantages is the fact that it is difficult to predict the end result of the product, as with ALM we have to keep in mind the product life from the original idea to the final stage of its life.

- **Agile**

The term agile software development refers to group of software development methodologies focused around iterative and incremental development, where
requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. There are several different agile development methods, which all have evolved from the so called Agile Manifest, and promote teamwork, collaboration, process adaptation as central practices through the software life-cycle (Wikipedia, 2011c). Agile model’s advantages are “adaptability, better code quality and reduced risks” (Vitikka, 2009). The disadvantages include the risks of frequent changes during the development of a product. The products that necessitate undergoing changes within the short time periods, would thus benefit from the agile technique.

Agile life cycle greatly considers the user input, and is companies that develop their products according to today’s customers’ wishes, rather than necessities. The product may have a short life, but is less costly for those companies that prefer to change their directions radically.

**V-Model**

The V-Model is another SDLC model, which is ofthen considered an extension of the Waterfall model. It’s name is due to the fact, that instead of moving in a linear from start to end, the process steps go down until the coding phase, after which they are bent upward, in such a way forming a typical V shape. The model shows the relationships between each phase of the development life cycle and its associated phase of testing. The horizontal and vertical axes represents time or project completeness (left-to-right) and level of abstraction, respectively (Wikipedia, 2011d). Figure 9 graphically illustrates the V-Model.

Some of the major advantages of the V-Model are it’s tailorability to the particular situation of an organization and it’s support for a wide range of development methodologies, including object-oriented, as well as structured analysis and design.
3.1.2 IT Maintenance and Operations

One of the hiccups of adapting ALM occurs because of the natural gap between pure software development and business strategies. Therefore when viewing ALM from the perspective of SDLC, we must not overlook ALM’s business standpoint.

The IT maintenance, also referred to as service management and ALM are closely related. According to Rossberg, “ITIL focuses on ALM from both perspectives as it stresses that we need alignment between both of these views in every phase” (Rossberg, 2008).

ALM, as seen from the perspective of the Service Management, focuses largely on “the deployment of the system into the production environment” (Rossberg, 2008). After application is set up, the operations it undergoes include bug fixes and newly approved modifications.

Information Technology Infrastructure Library (ITIL) “gives a detailed description of a number of important IT practices, with comprehensive checklists, tasks, procedures and responsibilities which can be tailored to any IT organization” (Bon, 2004).

According to van Bon, “ITIL is by far the best-known description of IT Service Management” (Bon, 2004). The ITIL comprises in itself several processes that need to be taken into account while running the IT organization, such as the quality of services from the users’ viewpoint, the management of employees within organization, and the management of processes in the company, which all need to be supported by an comprehensive ALM solution.

According to van Bon et al., the improving the quality happens through the scheme of Plan-Do-Check-Act.

The planning involves how the activity should be done, when and by whom. The doing refers to the implementation of the planned activities. Checking refers to testing of the implementation. Acting implies adjusting of the activities after the check.

This model is recursive, i.e. is recommended to be used continuously in order to ensure the good quality, as can be seen from the diagram below.
The ITIL is responsible for operating and managing of the organisation through several stages or processes, starting from its vision, and all the way to planning, and implementation. The scheme is similar to the above, although not recursive. First, each stage is planned, later it is implemented, then tested, and adjusted if necessary. Once this stage/process is completed, the next stage is being processed in a similar fashion.

The management of employees within organisation depends on the type of organisation. According to (Bon, 2004), it is not recommended to make a product-focused organisation, as it solely relies on the making of the product. Once an employee becomes a professional, he is likely to leave an organisation if the main focus is on the customer. The recruitment of a new employee may be a lengthy and costly process. Recruiting new employees each time the old ones leave is not a good practice, therefore it is better to ensure that the employee does not leave the organization once he becomes a specialist. There are several methods to achieve this goal, and they form the Human-Resource Management.

According to Rossberg, ALM has 6 phases from the standpoint of ITIL, namely “Requirements, Design, Build, Deploy, Operate, Optimize”, the latter 3 being the phases of the Service Management (Rossberg, 2008).

3.1.3 Application Portfolio Management

According to (Bon, 2004), Application Portfolio Management (APM) “helps businesses to manage large, complex sets of applications, understanding the way they work together in sharing data and in supporting end-to-end business processes and their use of the IT infrastructure” (Bon, 2004). APM is a repeated process, as it implies the managing throughout the actual application lifecycle.

The portfolio itself is referred to as “an information system which stores the major application attributes and provides an architectural view of the relationships between applications”, and it “makes it possible to provide information about particular applications or suites of applications and understand the way applications relate to one another” (Bon, 2004) and is therefore an important activity in every large software organization, which produces several product. Therefor it is also an important part of ALM.
The aim of APM is thus to collect every application, a organization delivers and integrate it in the organization’s portfolio. Collecting of the application may also consist of different stages: the case the application is bought from another company; and the case when it is developed within the same company. Here, according to (Mitchell, 2006), the cost in accordance to the benefit over value should be calculated. Regardless of the choice of getting an application, it then must be integrated and checked thoroughly.

APM should be viewed as a continuous assessment of the application portfolio, and thus the process of assessing should be carried out cyclically, as shown on the following graphic:

![Application Portfolio Management Lifecycle](image)

Figure 11: Application Portfolio Management Lifecycle

Mitchell analyses five attributes that should be taken into account when the application is being assessed, namely technical quality, functional quality, and strategic value of an application (Mitchell, 2006). Kellerman and Löfgren divide strategic value into three sub-principles, namely business value, investment value, and management value (Kellerman & Löfgren, 2008).

Below is a summary of the principles for assessing an application in the view of APM.

![Hierarchical view of application assessment](image)

Figure 12: The hierarchical view of an application assessment in view of APM.
Application under APM undergoes three phases: Assessment, Evaluation, and Planning. The assessment phase involves the gathering of valuable information regarding the application within the portfolio according to the principles, mentioned above. Then this information is used in the evaluation phase of an application. In regard to the application portfolio the evaluation is carried out, based on the biggest errors found within it, and how the application affects/is affected by them. In the planning, the actions to be taken in order to improve and optimise the application’s value are discussed; the solutions are proposed and calculated based on the cost and effectiveness.

3.2 ALM Tool Integration and coping with complexity in software development

One of the most important challenges many software companies are facing is dealing with the steadily increasing and getting out of hand complexity and rapidly shortening time to market requirements of today’s software systems. The increased complexity leads to more requirements, more complex architecture, writing code increases, inspection, validation, and testing increase. The necessity for holistically controlling these different aspects increases, however today’s disconnected (or at best fragile pieced together) different tools and systems with fragmented information, disconnected sources and often incompatible work modes are not designed to help to achieve this. This hinders a successful development process, as entropy gets in the way. The consequence is that software systems often cost too much, fail too often, and do not meet the basic business and architectural requirements posed by the users. Coping with this challenges requires a “continuous efforts to identify and eliminate complications that add no value. Management has to analyze the company and its environment to flush out the hidden linkages between costs, activities, and the decisions that generated them” (Jagersma, 2008).

Increased complexity, on the other side leads to more cross-talk, noise, and other sources of transient errors during normal operation. Today software developers “spend a large proportion of their time communicating” (Herbsleb, Mockus, Finholt, & Grinter, 2001). Indubitably, this affects the development of software to a great extent. “Coordinating product design decisions requires communication among the engineers making those decisions” is highly important, and is one of the priorities for organization to make feasible (Cataldo, M. Bass, Herbsleb, & L. Bass, 2006). It is virtually impossible for each employee to be able to communicate with every other employee within a relatively large organization. Therefore it is recommended that the “products must be split into components, with limited technical dependencies among them, and each component assigned to a single team” (Cataldo et al., 2006). The need for any modifications to the software, particularly to the code, may create the situation, where different teams will have to communicate with each other, and combine forces to find the solution.

ALM can be the solutions to the complexity problems, as independently from the taken perspective it has three main pillars (Schwaber, 2006):

- **Traceability of relationships between artifacts** - This is often a mandatory process dictated by compliance requirements. For many companies however an end-to-end traceability is creating and maintained to a large extend manually.
- **Automation of high-level processes** - Paper-based approval and handover processes are very common, however also very cumbersome, error-prone, costly and time consuming. ALM improves their efficiency by governing the entire process, automating these handoffs and by storing all associated documentation.
- **Reporting to increase visibility** - It is common today, that important stakeholders have only limited visibility on the project progress due to the manual effort required
to hold weekly status meetings, create progress reports, demonstrations, etc. This hinders the decision making process, especially in large scale projects and makes the determination of process improvements areas more difficult. ALM helps to alleviate this by providing automated reporting capabilities.

As the reader might have noticed these pillars are already well established today, however in their current form they are labor-intensive and error-prone. Today’s tools can help to create an maintain traceability, automate tasks and provide reporting capabilities, however they are restricted to a certain phase of the development lifecycle and the artifacts created during this phase (e.g. RMDBs, Test Management Tools, etc.). This creates isolated information silos, which require a lot of manual effort to be able to exchange information. This effort naturally increases with increased complexity of the product. This is how ALM works in its current form, often referred to as ALM 1.0 (Schwaber, 2006). With such isolated solutions, however, the vision of improving the efficiency of each of the three pillars of ALM towards creating an ecosystem of integrated processes and domain technologies, which can help to better cope with complexity and to reduce development time, can be considered a Utopia. Only by integrating the different life-cycle steps, correlating the different produced artifacts and providing unification and automation for each of the major participating roles and stakeholders (Carrillo & McKorkle, 2008), ALM can be brought to a whole new level and can put the vision of

- Increasing the consistency, predictability and measurability of the software development process by making common services available across the entire lifecycle
- Lowering the costs by reducing management effort
- Increasing the quality (independently of the project/product size) by helping a team streamline and adhere to processes, and finally
- Ensuring that available knowledge is found faster and not forgotten again within our grasps.

In regards to such an integrated ALM solution the author argues that, by putting everyone involved in the process onto the same system, regardless of their current location or role and ensuring repository neutrality, it will improve a teams ability to collaborate in-context of the work they are doing, to exchange and manage knowledge and information and to respond to changes in requirements, processes, etc. It additionally will provide the necessary end to end traceability and required visibility into the development processes and consequently give answers to the critical questions of “who is doing what, how they are doing it, how long it takes, and when goals are being met or missed” allowing in such a way to eliminates any wasted duplication of efforts and to help to address the complexity and time-to-market requirements of today’s software projects/systems (Magid, 2007).

### 3.3 Basic concepts and techniques of tool integration

According to (Ian Thomas & B. A. Nejmeh, 1992), “integration is not a property of a single tool, but of its relationships with other elements in the environment, chiefly other tools, a platform, and a process”, and “integration means that things function as members of a coherent whole” (Magid, 2007).

The tool integration can be done, i.e. the tools can be combined together to achieve the agreement upon a process, but many factors must be taken into account, for example such attributes as “data format, user-interface conventions, use of common functions, or other aspects of tool construction” (Ian Thomas & B. A. Nejmeh, 1992).

There are three different levels at which the product developers are working, namely the hardware level integration (involves the regulating of interfaces); software level integration (involves the regulating of interfaces); model level integration (‘models of system parts are developed and integrated throughout the development,
hence enabling incremental development and also software level integration in industry practices”) (Shi, 2007). ALM concentrates mainly on the last two level and can be combined with PLM tools (e.g. TeamCenter) to enable the integration between software development and system development tools. Although this is beyond the scope of this work, designing an industrial ALM Tool Integration solution, which is open and supports the integration of any tool as needed would indirectly also allow for integration of PLM tools and therefore could be easily extended to cover all three aspects of integration necessary for an end-to-end product development.

Until recently ALM was a non-topic, and tool integration was largely based on point-to-point integrations between tools, which did not provide the needed results. As already briefly described in the Introduction chapter, such integration were and still are based on the works of Earl (Earl, 1989) and Wasserman (A. Wasserman, 1989), and revolve around the concepts of data integration, control integration, process integration, and presentation integration.

Additionally to these four levels of integration, in theory there are also five types of integration, according to (Ian Thomas & B. A. Nejmeh, 1992):

1. Platform is linked with the services provided by platform;
2. Presentation links with user interaction through the Graphical User Interface;
3. Data, is linked with how the data is managed and used by tools;
4. Control is linked with interoperation between tools;
5. Process is concerned with the process of software development

All these makes it vital to be able to test how well one tool integrates with another. For instance, within the Data integration, we might have one tool that uses the stored data in a completely different way than another tool. In this case, the data records must be in a state so that both tools may be used to work with it. Now, imagine, there is a third tool that requires addition of an attribute to the data table. Changing the data entries may make both previous tools either not deliver, or deliver wrong results. The tool integration is only successful when each tool takes the other tool into account, viz. specifies its actions, effects, etc to another tool.

The following are the properties of Data tool integration:

- Interoperability – when both tools view the data evenly all together;
- Non-redundancy – when tools have little duplicate data;
- Data consistency – each tool must maintain the list of effects it has on data that is used by another tool;
- Data exchange – both tools must be able to use data;
- Synchronization – when “all the changes to all shared non-persistent data made by one tool are communicated to the other” (Ian Thomas & B. A. Nejmeh, 1992).

The Presentation of tools is highly important as well. For instance, users of any application tend to like another application of it has the same look and feel. Same could be said for tools. This attribute of looking and feeling the same when it comes to tools Thomas and Nejmeh refer to as “appearance and behaviour”, and determine the similarity of “the tools’ screen appearance and interaction behavior” (Ian Thomas & B. A. Nejmeh, 1992). The other property is what Thomas and Nejmeh refer to as “interaction paradigm”, and here the ability of learning perspective of a new tool is judged in accordance to the already learnt old tool. This property is not the same as the previous one, since this is no longer only about the recognizable appearance of a tool, but about the possibility of the user to get used to a new tool with little or no learning, by applying the experience with the old tool.

Control Integration refers to the tools’ abilities to share functions. Here we are looking at two main attributes for a tool to be identified as well-integrated. The provision refers to the property of a tool to offer services other tools may use. And the use of those services refers to the property of a tool that uses those functions accordingly.
Process Integration poses many aspects to definitions and attributes for tools, for it judges the set of tools as to how well the tools cooperating towards highest possible efficiency for the overall process. There are many conditions the tool must satisfy in order to contribute to the process as a whole, and at the same time be relevantly performing while supporting other tools.

In order to name the properties the tools must exhibit in order to prove satisfactory, several terms must be defined.

The process step is a resultant stage of work, taken towards the completion of the process. A process step integration property refers to how well the different tools cooperate to achieve the process step. The tool is therefore wrongly integrated if the resultant step is not suitable and/or adequate.

According to Thomas and Nejmeh, a “process event is a condition that arises during a process step that may result in the execution of an associated action” (Ian Thomas & B. A. Nejmeh, 1992). The property of process event integration refers to how well the cooperating tools compromise upon the choice of events, in the right order, so as to support a process.

And a process constraint is set to limit the process in some particular way. In this case, the tools are looked at from a perspective of how well they collaborate in order to “enforce a constraint” (Ian Thomas & B. A. Nejmeh, 1992).

So far, it is made clear that tool integration require a set of strict pre-defined conditions for the process and its sub-processes. But certainly it is impossible to bear in mind all of the specifics for each tool, and thus tool integration involves careful consideration and designing of tools’ integration with one another. This not only turns out to be costly, but requires experience and deep participation in the project of the personnel. The tool integration may consume a lot of time, which may prove great inconvenience for a company.

3.4 The ALM approach to tool integration

As already discussed in the previous chapter ALM was a non-topic until recent years, and tool integration was largely based on point-to-point integrations between tools. In recent years, the increasing globalization has made the already existing challenges of tool integration, more urgent than ever, as in a multi location environment, as well as outsourced development environment, it is important that a requirements management tool based in one geographical location be able to communicate with a Test tool based in a completely different geographical location.

Organizationally, this new phenomenon brings several challenges and opportunities for both - the involved developer teams and for management. On the one side, global software development (GSD) can help to overcome shortage in local IT skills, to achieve cost savings (e.g. outsourcing to “low wages countries”) and competitive advantage (e.g. “follow-the-sun-development”), increased quality (e.g. “round-the-clock-service”), closer proximity to markets and customers, etc. In order for an organization to be able to reap the benefits of this mode of work, however, it first needs to understand the challenges posed by increased geographical distance, time separation and cultural differences, which among other include loss of control and coordination, as well as communication problems, as global software projects require even more careful planning, collaboration, cooperation, on-going teamwork and better tools and mechanisms to support developers and managers in information sharing, knowledge management and ultimately in ensuring as few integration problems as possible (D. E. Damian & Zowghi, 2002; Lanubile, D. Damian, & Oppenheimer, 2003; Lehtonen, 2009; Tanner, 2009). All those challenges mainly revolve around few major issues, which make the problematic of tool integration, more important than ever, and yet more problematic than ever, as in times of globalization and global software development success more that ever depends on seamless, real-time data and information availability and visibility and cannot be encumbered by synchronization.
delays and process inconsistencies across geographical or functional boundaries. In this context, technical issues represent yet another problem, associated with global software development, and refer to a concerns on transparency, consistency and interchangeability of information and data, as well as missing support for distributed processes.

This has made both tool users and tool vendors realize that the point-to-point integration architecture without a centralized repository and/or framework and where the logic is hard coded in the integration codes, which has been predominant until now has been patchy at best, as it does not allow end-to-end automation, reporting and there is no way to visualize and manipulate traceability relations of three or more artifacts in a single interface (Basu, 2010). Such integrations are adequate only for small numbers of integration endpoints and typically create more complexity in developing and managing tools than they solve. The number of point-to-point integrations drastically increases and administration and maintenance become complicated for both customers and vendors, and accordingly the risk of error increases significantly. For example let’s take an integration of 10 different tools, each covering one of the following major software development areas (Requirements Management, Change and Configuration Management, Version Control, Modeling, IDE for Software Development, Defect Tracking, Quality Assurance, Deployment, Portfolio Management, Security). Due to the fact that the number of possible combinations of a set of n integration points taken 2 at a time is given by

\[(n!) / (2! (n - 2)!)]

it means that trying to integrate those 10 tools towards an end-to-end ALM using a point-to-point architecture, would require 45 unique integration, which is a solution that does not scale (The-Eclipse-Foundation, 2008).

This has led to the emergence of the integrated ALM, which is a new approach to tool integration aimed at achieving comprehensive process control and coordination across all software development activities by integrating the different product and tool silos, currently in use across the lifecycle. Instead of using the, until this time common, fragile point-to-point integrations however, ALM takes a broader, more holistic approach to the application lifecycle and focuses on driving better business outcomes, by introducing some sort of central negotiator that manages interactions between applications or an intermediate communication format to prevents having to integrate an application several times with several other applications, but to rather carry out one integration on the central node (The-Eclipse-Foundation, 2008). Different vendors apply here different technological means to achieve this, as it can be seen in the analysis of existing ALM tools in Chapter 4.5, however they all provide deeper semantic integration than the point-to-point architecture. Unfortunately this often comes at the cost of not being able to select which lifecycle tools are integrated with each other to form an integrated end-to-end ALM, but having this choice made for one by the tool vendors. The latter usually integrate their own tool with each other and offer no support for third parity tools and ALM stacks, which leads to the old problem of having to use fragile point-to-point integration when wanting to integrate a new tool in the existing ALM tool chain. Although an improvement compared to the initial situation of using solely point-to-point integrations the author believes, that an ALM solution can offer both, deep semantic integration and openness and interoperability, which allows to integrate on demand best-of-breed tool toward an ALM solution.

Therefore the next chapter discusses in more detail the current form of ALM with the just described advantaged and disadvantages, and provides an answer to the question, why in the authors opinion further ALM research is necessary. Chapter 4 after that initiates this further research by addressing the question of what such an improved ALM solution should offer to its users and also provides initial ideas on how this can be achieved.
4 APPLICATION LIFECYCLE MANAGEMENT TODAY

In this chapter we shall discuss what ALM presents in itself currently, and its roots in the past and present approaches from various fields. ALM has both advantages and disadvantages, and this chapter talks about what they are, and why we need a further research in the area of ALM with regard to its disadvantages and current state.

4.1 What is ALM today and how does it relate to lifecycle approaches from other fields

Today ALM occupies an important role in the industry, as well as in the research. ALM products already provide big and medium companies with their services. As “cooperative tools are able to achieve a greater efficiency”, and ALM products offer a wide range of tools to choose from, thus ALM had become irreplaceable for many organisations (Kiper, 1988).

ALM does not gain its success from the brand new strategies. ALM is often compared to the Product Lifecycle Management (PLM) of the mechanical/electronic product. The reason why the PLM should not be used for software development, is because there are different criteria and market for physical and software products. The lifecycle of PLM is similar in definition, to be precise the PLM’s “period usually consists of five major steps or phases: Product development, Product introduction, Product growth, Product maturity and finally Product decline” (Komninos, 2002).

The production of hardware is a different process as a whole, despite the similar production schedule. The developers must at the stage of original idea, before designing, calculate the price of the material for the product. Since software is not a physical device, this step can be omitted. At the step of designing, different attributes are taken into account – in case of hardware, it is often that the developers design towards the compact solution in terms of physical size, and yet offer big size in terms of device’s capacity. For example a relatively small USB memory card has a size of 16GB in 2010, which is the size of the hard disk memory of an average laptop in 2001. With software, the design takes on different attributes. For instance, software is almost always designed with the possibility of update in the near future. While hardware is steady, software needs to be constantly tested, updated and protected from viruses, malware, spyware, etc. Software may always be changed, modified, edited. Hardware on the other hand may not be modified, and in fact it heavily depends on the materials it is made of, for they wear out with time.

For the past decade, the software became so diverse, offering so many possibilities of services, that the software development and production have become in need of software-specialised management. Yet, the solution is still quite costly, and ALM needs to be further developed.

Figure 13: Software development timeline (Novulo, 2011)
According to Novulo, “with Integrated Development Environments (IDEs) and Application Lifecycle Management (ALM) suites, the trend has been set towards increased and improved involvement of project stakeholders” (Novulo, 2011).

The reason why ALM emerged out of the previously done point-to-point tool integrations is because there was a need for the organizations to connect different stages of managing an application together. ALM comprises different point-to-point integrations for bringing up the services and tools together, that are to assist the application lifecycle. The reason why traditional practices for other areas of produce (not software), such as Product Lifecycle Management (PLM), did not work out for the application managing processes, is because the software development is vastly developing with each passing year, and now requires specific tools for managing it.

PLM “is the process of managing the entire lifecycle of a product from its conception, through design and manufacture, to service and disposal” (Wikipedia, 2011e).

ALM is similar in idea, but specific for the software product. The software applications have several problems that are specific to them. Below are some of the most typical ones:

- The software has a high competition on the software market
- The development may take a short time if managed properly, but otherwise it may take a very long time. The bugs can be extremely difficult to fix.
- Customer is picky, as he pays the money for something he cannot understand in its entirety
- The code, design, idea, etc. is also a subject to being duplicated by another software development company
- It may be difficult for the software product to gain the popularity eventually – the whole project may prove unprofitable
- The product is subject to virus attacks

In ALM the actual application development is a very complex stage, which follows after the design and approval of business case, and is radically different from the product manufacturing stage of PLM – realization, which also follows after the design. Depending on the products, the application development stage may be just as costly or even more expensive than the product realization stage of PLM. PLM is more concerned about such issues, as quality manufacturing and product delivery. Therefore the traditional PLM processes do not correspond to the needs of the managing of the software application. But the original idea of PLM, that the product is to be managed through all of the stages in its lifecycle, and all of these stages are present and need to be connected, was adapted.

Thus, the application lifecycle management’s main task is to take the application through its entire cycle, providing three important services: “traceability of relationships between artefacts, “automation of high-level processes”, “providing the visibility into the progress of development efforts” (Schwaber, 2006).

ALM transpired because of the need of guiding the application through its every stage. The figure below shows these stages, as well as the sequence of ALM activities:
As seen from the figure, the business (or governance), development and operations are closely connected, and unless the application’s time is up, the set of activities continue in a circular iterative fashion.

Chappell claims “doing [governance, development, and operations] well is a requirement for any organization that aspires to maximize the business value of custom software” (Chappell, 2008).

The smooth transformation between the activities in the above three spheres is required, but is difficult to organize and maintain.

According to Chappell, the “project management tools should be connected to development tools, which in turn should have connections to the tools used for operations” (Chappell, 2008).

ALM has many aspects, with activities specific to a particular organisation, and to make all the puzzle pieces come together, there is a need for integrating the tools of different fields. Below are the main constituents of ALM, necessary for the development and maintaining of any software application:

As can be seen from the figure above, ALM emerges from other lifecycle approaches in different spheres. ALM consists of various types of management activities that provide guidance at different time span of the application.
There are six phases of ALM, in accordance to ITIL:

- **Requirements** – the gathering and documenting the requirements for the future application
- **Design** – a design blueprint for an application is created.
- **Build and Test** – “code, configure, compile, and test the application” (Bhamidipati, 2010).
- **Deploy** – after testing and/or updating, the application can be released.
- **Operate** – “manage the application in production using monitoring tools to ensure that it performs as expected” (Bhamidipati, 2010).
- **Optimize** – check the application for efficiency, and introduce feasible improvements.

The different managerial processes govern these phases.

Chappell states, “In ALM, the purpose of governance is to make sure the application always provides what the business needs” (Chappell, 2008).

During the governance, the initial idea about what application is going to be is first stated. Several alternatives are proposed as for the initial idea, and the most convenient one is chosen.

According to Chappell, “the first step in ALM governance is business case development”, after which “application development starts, and governance is […] implemented through project portfolio management” (Chappell, 2008). In ALM, Application Portfolio Management provides “an ongoing understanding of its benefits and costs”, and the “governance for the deployed application, addressing things such as when updates and larger revisions make business sense” (Chappell, 2008). The APM is a vital part of the governance period, as many hiccups may go unnoticed, that may cause great instabilities in the future.

ALM would not be fulfilled in terms of software delivery without “its connections to the separate but related disciplines of project portfolio management (PPM) and IT operations” (Schwaber, 2006).

At the same time, i.e. after the business case approval, the Software Development Lifecycle starts, when the actual application is being developed. Note that in Figure above, the Software Development Process is another name for Software Development Lifecycle (SDLC). The SDLC is an important subset of ALM, where the delivery of an actual application is managed.

Software development lifecycle, as explained in detail in Chapter 2 of this thesis, may have different models, most often implemented are Waterfall, V-Model, Spiral, Agile.

Depending on the determination and objectives of the organisation, the three different approaches may be chosen for the stage of SDLC. It is recommended to use agile or spiral ALM approaches, if the application after deployment is planned to be developed further. If the deployment is the last stage in application lifecycle, the waterfall ALM approach may be less costly and more convenient, depending on the circumstances of the business case.

Szalvay explains in his article, that “unlike the "linear" waterfall method, Agile teams use an "iterative" or "inspect and adapt" approach to ALM to address the changing requirements, complexities and risk factors that will arise over the course of the software project” (Szalvay, 2010).

On comparing the spiral and agile approaches, Vitikka states that “most agile models aim at reducing risk by shortening the development time” (Vitikka, 2009). Through agile approach it is also easier to adapt the project to changes, than it is with the spiral approach. Agile approach has, however, disadvantages to it as well. Therefore when choosing the right ALM, it is important to consider the specifics of the organization.
The next stage is the deployment of application, and it normally should not be planned as the last stage, unless the application is initially arranged to be of a very short-time use.

Once the application is developed, it is deployed, and usually then it is being updated/upgraded/improved over time.

Looking back at the Figure B, it can be seen that “as with Governance, the Operations [are] intimately connected to the Development” (Chappell, 2008).

According to Chappell, “once the application is deployed, it must be monitored throughout its lifetime” (Chappell, 2008).

The problem is that a lot of organisations overlook the fact that they have to take into account the application’s after-deployment life. Many organisations, according to (de Jong, 2011)), fail to recognise and plan for the application’s end of life. This results in the non-realistic investment in the application. If the goal of the application is to last for a longer period of time, although it is not feasible, the financial resources may be spent in vain, aiming at a bigger project. On the other hand, if the application is overly criticised and planned for a shorter application lifespan, say once it has been deployed the application was not planned to be maintained further, then it may be difficult to develop and maintain it further.

Thus, it is of high importance that the application’s lifecycle is realistically estimated.

After the deployment stage, and provided the application was planned to be further maintained and developed, the operations stage comes next, which involves such activities as the monitoring, developing and supporting of an application. It is a continuous process that will at some point in time lead to the pick of the application’s success, and eventually smoothly lead to a decline of the application and the termination of its lifecycle. The Operations stage becomes tied figuratively to the original design of the application. The development activities for the application would be dependent on the original design, coding, etc. in terms of factors such as finances, editing, design improving, etc. The possible developments must be discussed at the very early and first stage of business case. If the developments are approved for the possibility of future implementation, then the initial design of the application would be greatly taken into account with the consideration of further developments. The maintenance and support of the Operations activities are very important, as the customer may choose another subscription, if those activities are not carried out in the systematic and continual manner.

4.2 What are the advantages with the current form of ALM

ALM has become so interesting for many organizations, due to the many advantages it offers, including, but not limited to:

- ALM keeps all of the application lifecycle activities in sync, rather than just focusing on the technical side of the application development stage.
- “Development effort can still fail miserably even if analysts document business requirements perfectly, architects build flawless models, developers write defect-free code, and testers execute thousands of tests. ALM ensures the coordination of these activities, which keeps practitioners' efforts directed at delivering applications that meet business needs” (Schwaber, 2006).
- ALM promises the possibility of cost reductions within the organization, where it is used appropriately.
- ALM has been known to increase the rates of sales for businesses, and decrease the costs of processes within organizations.
- ALM “increases flexibility by reducing the time it takes to build and adapt applications that support new business initiatives” (Wikipedia, 2011f).
• ALM enables stakeholders to actively participate as they have an access to the observing and tracing of ALM’s discipline within the organization.
• ALM is said to break “boundaries through collaboration and smooth information flow” (Wikipedia, 2011f).
• Today the current integrated ALM, in comparison to other practices, “can connect in a better way allowing teams, individuals and organization to gain better insight into the application development process” (P. Brown, 2010).
• The project management is improved in the organization.
• The overall quality of the product is improved.
• ALM improves the software quality, making the deployed application desirable to its customers in terms of outcome production.
• ALM “increases productivity, as the team shares best practices for development and deployment, and developers need focus only on current business requirements” (Wikipedia, 2011f).
• “Implementing the Application Lifecycle Management Suite can reduce the number of opportunities for errors in the product development lifecycle and provide opportunities for enforcing business process rules” (Sittler, 2008).
• The successful integration of ALM tools allows for a faster development
• ALM helps to cut “maintenance time by synchronizing application and design” (Wikipedia, 2011f).
• Saving the financial resources on managerial matters, ALM allows for these resources to be invested in such matters as human competence and up-to-date technologies.

4.3 What are the disadvantages with the current form of ALM

Despite all the advantages, in its current form ALM also has several important shortcomings:
• According to Schwaber, most of ALM’s solutions were primarily built up by constant additions, with no initially planned design, and “the dominant structure of today's application lifecycle management solutions is tool-to-tool integration, and this integration is never as deep or resilient as advertised - especially when it's integration of different vendors' tools” (Schwaber, 2006).
• The integration of tools is individual for each project, and the most frequent problem occurs when one tool is changed/modified, the other tools become in need of modification.
• “Redundant and inconsistent ALM features locked in practitioner tools causes Lack of cross-life-cycle transparency” (Schwaber, 2006).
• Tools are tricky and costly to upgrade. The application’s cost for the duration of its whole lifecycle is greater.
• ALM vendors offer role-based tools, meaning that each tool can carry out one role/function. The biggest problem is that each organization has its particulars, and the tools offered by vendors do not cover up for every necessary function. Thus, there happens a condition referred to as vendor lock-in, when “the IT organization has to choose between changing its roles, licensing multiple tools for a single role, or purchasing more features than a given role is likely to need” (Schwaber, 2006). That’s what makes these tools “complex and expensive” with “more functionality than any one individual is likely to need” (Schwaber, 2006).
• Role-based tools may result in the low productivity, and depleted efficacy.
• “Role misalignment leads to overspending on ALM licensing” (Schwaber, 2006).
• Lifecycle tools “feature an impressive amount of redundant and usually inconsistent functionality in areas like workflow, collaboration, reporting and analytics, and security” (Schwaber, 2006).
• The “functionality that would be valuable to the entire development team is often available only from within a single practitioner tool” (Schwaber, 2006).
• Collaboration problems between stakeholders based on cultural and language problems.
• Complicated for a stakeholder to give an account on the status of the application process. To know additional information, a stakeholder would need to turn to other tools, and “since each tool has its own internal data, [the stakeholder] would first need to consolidate and normalize the data” (Shaw, 2007).
• The activities of one software development lifecycle are usually inter-related with the activities of another software development lifecycle, particularly since the processes are carried out continuously throughout all the phases of the application’s life. Yet each of the set of tools targets only its very specific development lifecycle phase, and in order to successfully coordinate and control the software development activities, the artefacts and objects produced by these tools need to be interlinked and traced among each other for the purpose of process automation, reporting, impact analysis, regression testing, etc. and to help administer a cohesive and comprehensive software development process (Kääriäinen, 2008; 2011; Kääriäinen et al., 2009).
• According to Shaw, “with traditional ALM tools, traceability and reporting across disciplines is extremely difficult” (Shaw, 2007).
• “Repository synchronization is the primary means for integrating application lifecycle tools today”, and “it is often difficult to establish, costly to maintain, or flat-out unworkable” (Schwaber, 2006).
• “Integration through brittle repository synchronization mechanisms leads to Process assets are opaque and difficult to manage” (Schwaber, 2006).
• Data may get stored in an unstructured manner when trying to build and keep up the synchronizations.

4.4 Why do we need a further research in the area of ALM

The today’s market of software is emerging rapidly, and the software companies are often in a hurry to produce new application, while overpaying for the tools, necessary for its development. During the last decade, so many changes happened in the field of computer science, that there has not been enough time to carefully research the area of software development. The competition, responsibilities and race after innovations within organisations are growing at high rates, and with them the so are the risks of infeasibility of all those projects. One wrong step taken may lead to bankruptcy. The industry does not allow for a break. At all possible costs the companies try to reach their goal of introducing the ready bug-free application to the market, before their competitors. Some companies even turn to unethical ways of software development: they may allow for the illegal use of copyright tools, for modifying other companies’ programming code, etc. All of these problems originate from the inability to organise the employees to realistically follow the time schedule from the very original idea to the development, implementation, growth and finally the decline of software application. Most employers like to see the period of the actual coding in process. They pay little attention to the fact that even the good code does not eventually cover up for other aspects of the software standards.

The Eclipse Foundation has stated that the “point-to-point integrations are adequate only for small numbers of integration endpoints and typically create more complexity in developing and managing tools than they solve. Given the myriad of products, tools, and services that are typically integrated to manage an application change from its inception to fulfilment, the number of point-to-point integrations drastically increases and administration and maintenance become complicated for both
customers and vendors alike, with the risk of error increasing significantly” (The-Eclipse-Foundation, 2008).

Consequently, this research is necessary because ALM has quite limited features, and presents a very raw solution for the business of today.

ALM is missing out many pieces for it to become a true solution for managerial purposes within organisations. For example, “technologies such as application security and virtualized testing, among others, remain largely outside the ALM process today, even though they are widely recognized to play important roles in how applications are developed and managed” (de Jong, 2011).

Jama states that ALM 1.0 is “about the introduction of specialized tools and the ability to manage data (requirements, tasks, code, defects, etc)”; ALM 2.0 is about the adapting of the agile techniques; and “the next evolution of ALM will focus on people” (Blog, 2010).

The Agile ALM is said to be the main idea of the next generation ALM. There are many controversies and contradictions to this idea, and the practical side behind this idea. According to de Jong, many see the Agile ALM as a natural evolution. And what is currently being done in order to reach the status of Agile ALM, “ALM tool makers are adding support for agile practices including short iterations (where the team delivers working software every one to three weeks); continuous integration (where team members integrate their work at least daily); refactoring (making code more efficient without changing its behaviour); user stories (high-level definition of a requirement); and backlogs (list of deliverables ranked according to business value)” (de Jong, 2011).

Contrariwise the vendors, who claim to introduce and implement the Agile ALM, are believed by some to advertise the new tools and packages, the real contents of which would be found out only in practice.

Kersten believes that “the continuous integration and delivery loop, becoming increasingly popular in Agile deployments, will put a new set of requirements on both ALM tools and the connection between the running application and the developer’s workbench” (Kersten, 2011).
Because the deployment and after-deployment stages have been largely ignored for a long time, Kersten makes several suggestions and advices, and claims that “vendors that tie together the programming model, frameworks, virtualization, hosting and ALM will be the winners in the battle for developer mindshare” (Kersten, 2011).

Figure 17: Towards successful deployment (Kersten, 2011).

The figure above is suggested with the thought in mind that for this scheme to work, “the IDE tools must be capable of be effective in displaying that information alongside the code”, and “at each step of the way, it’s the ALM tools that will facilitate communication between the various stakeholders in the process” (Kersten, 2011).

Kersten concludes, saying that “connecting the IDE to an ALM hub and using the hosted source and builds to drive deployment will provide a convenient on-ramp to application hosting offerings, and will significantly reduce the configuration and administrative burden currently placed on developers”; and adds that in order for this operations to be accomplished, the considerable changes are necessary “from ALM stacks, which have generally ignored the details of application build and deployment to date” (Kersten, 2011).

Many authors make assumptions at how ALM may be improved, but there is always a series of drastic and/or radical changes that need to be involved in order to bring their suggestions to life. Because ALM tools were added up for the duration of its existence, and not planned forward to have the quality of integration and working together, ALM tools resulted in a phase, when it is difficult to change their direction. It is also difficult to halt their further development, as the need arises sooner than all the errors can be fixed, and the structure redone in design and implementation.

And despite all, the organisations already set high hopes for the future ALM tools to help to assist their business matters. But the fact is that these organisations are already in need of these improvements, and the waiting time sets a burden for the successful and flexible software development techniques of lower costs and feasible time schedule, and thus the development of the new applications with new opportunities that they could offer us today.

The reasons why ALM needs to be improved are:

- According to Schwaber, there arise “higher levels of awareness among IT shops of ALM than of other emerging disciplines like information life-cycle management and Agile software development processes” (Schwaber, 2006).
- “Nearly one-third of enterprises are already using [some form of] ALM” (Schwaber, 2006).
- Customers only want to pay for the features they require, not for the expensive tools that have plenty of extra other features.
- Practitioners want to use the tools in a faster and easier way
• Practitioners require for the processes to “share common components” (Schwaber, 2006).
• Vendors want to be able to upgrade the tools in an easier and more reliable way
• According to Schwaber, “collaboration, workflow, security, and reporting and analytics”
  “should be available from within multiple practitioner tools” (Schwaber, 2006).
• Moving old assets in ALM 1.0 is a complex procedure.
• Practitioner tools do not provide the full communication of activities in ALM 1.0
• Customers and partners wish to “build deeper integrations with third-party tools”
  (Schwaber, 2006).
• There is a need to have for all parties a “better integration between point tools for
  requirements, coding and testing” (de Jong, 2011).
• To establish the connection between different tools, particularly from different vendors,
  the coding is necessary. Thus, linking the tools is costly and time consuming.
• According to Kersten there is “a new level of automation [...] needed to help us focus on
  delivering application features and business value in the face of increasing platform
  complexity” (Kersten, 2011).

To summarise, ALM needs improvements in the following areas: practitioner tools, “repository neutrality”, open integration, microprocesses, macroprocesses (Schwaber 2006, p.6). There is also room for improvement in other areas of ALM, since “the majority of today’s ALM solutions have grown through accretion rather than through purposeful design” (Schwaber, 2006).

Due to many contradictions in the area of ALM, and the lack of statistical data, there is little researching done in the area. Many ideas arise, and with them even more barriers. There needs to be a comprehensive summary done first, a deep and detailed survey carried out, and results put together before arriving at any premature suggestions and/or conclusions. The evolving of a new upgraded ALM is on the verge, and it will yet have plenty of room to further development and improvements. ALM is a relatively new area of practice, and a yet newer area for research. More and more tools appear at an increasing rate. But the clear structure, in terms of usability and efficiency, has already become a major requirement. Finding a solution for it through the means of an independent research is a prerequisite for the successful development of the future of ALM’s tools and platforms, for their expansion is inevitable.
5 ALM 2.0 – DESIGN FOR THE FUTURE

This chapter summarizes our research findings in a chronological order, according to the order of the research phases.

5.1 Identification of relevant ALM stakeholder (stakeholder analysis)

Based on an in-depth literature study and several informal interviews and talks with the people involved in the previously described project, as well as external users, customers and affiliated organizations, the authors identified the following stakeholders (and roles) of an integrated ALM solution and can be associated with a process phase, belonging to one of the three major views of ALM as defined in (Rossberg, 2008):

- **Application Architect**
  The application architect is responsible for the selection of the paradigm and technology for application program-to-program communication (APPC) among the components, for determining the overall priority ranking of each of the possible system qualities (cost, reusability, robustness, etc.) so the other architects can design models that enforce the “balance of concerns” and for defining the application tiers, frameworks, components types and interfaces (S. Wright, 2011).

- **Board Director**
  The Board director is responsible for ongoing governance of the entire organization.

- **Business Analyst**
  Business Analysts are responsible for identifying the business needs of their clients and stakeholders to help determine solutions to business problems. The Business Analyst is responsible for requirements development and requirements management. Specifically, the Business Analyst elicits, analyzes, validates and documents business, organizational and/or operational requirements (Barret, 2009).

- **CIO**
  The CIO is responsible for overall leadership, planning, development, and management of information technology resources across an organization.

- **Change Manager**
  The Change Manager is responsible for the success of the change management strategy and implementation for the project.

- **Configuration Management Engineer**
  The Configuration Management Engineer is responsible for software configuration and release management activities for all software.

- **Customer**
  Customer refers to a current or potential buyer or user of the products or services provided by the Organization.

- **Data Architect**
  Data Architect is responsible for the design, structure, organization, and maintenance of data for a particular company or project (Harper, 2011).

- **Database Administrator**
  The DBA is responsible for the design, implementation, maintenance and repair of an organization's database (Wikipedia, 2011g).

- **Developer**
The Developer role is responsible for designing and implementing the actual solution, based on the inputs from the requirements engineering and architecture related roles.

- **Enterprise Architect**
  The Enterprise Architect is responsible for ensuring the creation of successful enterprise solutions, which solve a customer’s business problems from inception through execution.

- **IT Director**
  The IT Director is responsible for proactively managing and implementing the company’s Information Technology and ensuring the integrity of its processes through long-term strategic goals and short-term tactical plans.

- **IT Manager**
  The IT Manager is responsible for strategic direction, planning and technical leadership in the organization.

- **Infrastructure Architect**
  The Infrastructure Architect is responsible for the creation of technology infrastructure roadmaps, making assertions about directions and determining their consequences for the infrastructure strategy and architectural approach. The key responsibilities of the Infrastructure Architect are to define secure infrastructure architecture, resolve high-level functional issues, and provide continuity in all major technology platform and solution design decisions (Borden Ladner Gervais LLP (BLG), 2011).

- **Network Architect**
  The Network Architect is responsible for overseeing network and data center design, installation, maintenance, and problem resolution. Additionally, the Network Architect installs, configures, and troubleshoots networks and associated components.

- **Operations Manager**
  The operations manager is responsible for overseeing, designing, and redesigning business operations in the production of the goods and/or services of the Organization.

- **Programmer**
  see Developer.

- **Quality Assurance Manager**
  QA Manager is responsible for both the resources and tools used to ensure that defects are identified early and do not materialize once the applications go into production (IPSUS, 2011).

- **Release Manager**
  The Release Manager is responsible for planning, scheduling and controlling the movement of Releases to test and live environments. His primary objective is to ensure that the integrity of the live environment is protected and that the correct components are released (IT Process Wiki, 2011).

- **Requirements Engineer**
  see Business Analyst.

- **Security Architect**
  The Security Architect is responsible for identifying current and emerging IT vulnerabilities, for championing security standards and processes, review security architecture, and plan, develop and implement IT Security remediation programs (Platform Solutions, 2010).

- **System Administrator**
  The System Administrator is responsible with installing, supporting, and maintaining computer systems, and planning for and responding to service outages and other problems. Additionally his duties may project management for systems-related projects, supervising or training computer operators, and being the consultant for computer problems beyond the knowledge of technical support staff (Wikipedia, 2011h).
• **System Analyst**
  A Systems Analyst researches problems, plans solutions, recommends software and systems, and coordinates development to meet business or other requirements.

• **System Architect**
  The System Architect is the high-level designer of a system to be implemented. The systems architect establishes the basic structure of the system, defining the essential core design features and elements that provide the framework for all that follows, and are the hardest to change later. The systems architect provides the engineering view of the users' vision for what the system needs to be and do, and the paths along which it must be able to evolve, and strives to maintain the integrity of that vision as it evolves during detailed design and implementation (EPS Corp, 2011), (Wikipedia, 2011).

• **Team Leader**
  A Team Leader is responsible for forming a team with the goal of creating a specific outcome, given set resources and a timescale

• **Test Analyst**
  A test analyst is responsible for decomposing test requirements, test case creation, test execution and assisting the test lead in defining the test strategy and the test plan (Testing Realms, 2011).

• **Test Developer**
  The test developer is responsible for defining, implementing and automating test cases to validate various features.

• **Test Manager**
  The test manager helps develop product test strategies, and provides test expertise to the testing group and gathers product information so that corporate management can decide when the product is ready to ship (Testing Realms, 2011).

• **Tester**
  A tester is responsible for evaluating the product that has been created by the developer in comparison to the goal defined by the designer. In the end, the software should match the conceptual design as envisioned by the designers, with no logic or performance errors (King, 1997).

• **User Acceptance Tester**
  User Acceptance Tester is responsible for executing user acceptance testing on the final product.

The so identified roles we used to create a stakeholder model, which can be seen in Appendix B.

### 5.2 Elicitation of future ALM requirements through stakeholder interviews and questionnaires

This section shows the questions used during the interviews and the survey, as well as the insights obtained through the analysis of the answers. A PDF-Export of the entire survey, as well as a summary of all received answers can be seen in Appendix C respectively Appendix D.

#### 5.2.1 Interview Questions

The following questions are based on a thorough literature review (Göthe, Pampino, Monson, & Nizami, 2008; Kääriäinen et al., 2009; Rossberg, 2008) (Kääriäinen, 2011; S.A. Architect, 2011) and are part of the survey and were used to guide several interviews, both undertaken with the primary focus on understanding an organization's practices and policies in order to learn its needs and requirements for an ALM solution that can offer a complete integration of the disparate tools (respectively a single tool replacing them) and will also ensure full traceability, real-time visibility.
into your project and application status, including costs and schedule variances and will support the project management goals.

- **General Questions** aimed at gaining general information about the interviewed person and its organization.
  - What role do you play in your organization?
  - Approximately how large is your organization?
  - Where are you based?

- **ALM Definitions** aimed at gaining understanding for the interviewed person’s and his organization's understanding of the ALM discipline and the ALM tools and their current ALM usage profile and experienced problems.
  - What is your definition / understanding of ALM, both as a discipline and as a product category?
  - How relevant / important is ALM for your organization?
  - How relevant / important is ALM for your particular role in the organization?
  - What particular improvement(s) are you hoping to achieve by applying an ALM solution in your project(s)?
  - Does your organization currently use an ALM solution?
  - Did your organization stumble upon any features in ALM that may or do postpone or slow down the product development?
  - Does your organization currently integrate different tools and components? If yes, which ones?
  - How easy is the integration to implement?
  - Are you satisfied with the results of the integration on a data and semantic level?
  - Process and method level?
  - Service and control level?
  - Do you use a simple point-to-point integration between the different tools or some common integration bus and shared information model?
  - Please list some of the problems you experienced during the integration setup phase?
  - Do you satisfied with the availability and performance of such standards and information models or do you believe that this area needs further research?

- **ALM Scope** aimed at finding out the appropriate scope of an ALM solution in the interviewed person’s opinion.
  - Which statement regarding the appropriate scope for an ALM solution do you agree most with?
  - An ALM solution should have solely an application (in terms of application development lifecycle) focus!
  - It is also (besides the application focus) important to integrate IT operations into an ALM solution, as this will allow resolving application (in terms of application development lifecycle) problems and to modify the process around the software delivery according to any changes made in the production environment.
  - An ALM solution must be flexible and support different levels of business involvement, as there is no "one size fits all" solution that is appropriate to all organizations.
  - What do you believe are key components and services, which an ALM solution necessarily needs to integrate for the management and governance of the application lifecycle process?
• Do you believe that ALM in all its facets is feasible for small and middle-sized companies?
• Do you believe that ALM in all its facets is necessary for small and middle-sized companies?

• **ALM Integration Level** aimed at finding out the appropriate level of integration of an ALM system.
  • Do you believe that for a tool to be considered a high-quality ALM tool it has to be a complete solution like for instance a suite offered by a single vendor? Or do you believe that it is better for it to be an orchestration and integration of several best-of-breed tools, each focused on the specific functions necessary in a particular lifecycle step?
  • If you were given the choice between the introduction of an single vendor ALM solution and a framework allowing deep best-of-breed tool integration, which solution would you prefer?
  • Using a complete solution, offered by a single vendor would involve the migration of all data from the different repositories and tools, your organization has already invested in, to the new ALM system. Is your organization prepared to cope with these new costs (financially in terms of new licenses and and in terms of time).
  • Where across the software delivery lifecycle would you like to see more process support (frameworks, best practices) from an ALM tool?

• **ALM Features** aimed at understanding the organization’s practices. The information is used to derive necessary features and capabilities of an ALM solution.
  • **Requirements Engineering / Product Management**
    • Do you during the requirements capture activities tend to mix different techniques depending on the nature of your requirements needs, written, use cases, diagrams, etc.?
    • Do you normally perform some kind of stakeholder analysis and prioritization in your projects and if yes how do you use this information later (e.g. prioritization of requirements, etc.)?
    • Do you agree that change auditing and notification should be very flexible and carefully controlled, as over-notification is sometimes worse than none at all, resulting in inattention to changes in general?
    • Do you use impact analysis to track the impact of changes in the requirements on the rest of the system?
    • Do you plan for reuse of requirements across the different product versions?
    • Do you explicitly manage variability in your system?
    • Do you maintain a tight correlation between the works you perform and the initial requirement to ensure that what’s delivered is what the user needed and wanted?
    • To what other artifact do you trace your requirements?
  • **Testing**
    • What different types of testing happen?
    • How is testing measured?
    • How is testing planned?
    • What test related roles are there in your organization?
    • Does your organization use test automation?
    • Do you consider code coverage?
    • Are your system tests directly traced to requirements artifacts? Do you use these traces to derive any metrics? (e.g. requirements covered by tests)
• Do you propagate risks and priorities identified during the RE activities to the according test artifacts?
• Based on what criteria do you plan your regression test suites?

**Agility & Formality**
• Are you processes rather flexible and therefore require small and lightweight tools with few restrictions and lots of auditing, or are your processes well tested and proven, and require a tool able to enforce them?
• Do all decisions need to be documented?
• Do individuals work on several projects concurrently?
• Are the project's teams geographically dispersed?
• Do the scope and requirements of the solution change a lot?
• Are your projects time driven?
• Do your projects have well defined roles & work streams?
• Do you need to accurately track the exact time spend on specific tasks?

**Architecture**
• Does the architecture consider flexibility for change?
• Code Quality
• Do you have a well-defined and thorough check-in process, which includes quality checks?

**User Interface**
• Do you use storyboarding?
• Does UI plan consider costs associated with different design options?
• Are user interface designs prototyped before they are implemented?
• Are changed in UI tracked against requirements and builds?

**Project Management**
• Is the project's vision and scope well defined and understood and does it map to the business problem?
• Do you think that the Customer needs visibility into the project in order to be able to effectively support it?
• What metrics do you derive from the project? (e.g. budget, schedule, risk, etc.)
• Are risks known and actively managed?
• Are the dependencies on third parities and the customer being managed?
• Are any specific methodologies and best practices being followed? Do you expect them to be supported out of the box by the ALM solution?
• Are resource plans in place and being managed in-line with project scope and budget changes?
• Does the development process rely on custom-developed tools?

**Configuration Management**
• Is a build process well defined?
• Is there a regular build schedule?
• Is an automated build verification process in place?
• Is there an effective build failure/success notification process in place?

**Other**
• Do you believe that there are specific aspects of automation that should be sought out when looking at today's ALM tools? Which ones?

**People**
• What roles need to be supported by the new ALM solution?
• Do you believe that using a single tool for all the different roles in the ALM can lead to low productivity, due to increased tool complexity?
• Do you believe that role misalignment can lead to overspending on ALM licensing?

5.2.2 Results of the elicitation

In April 2011, 45 people (the previously mentioned project team, as well as external users, customers and affiliated organizations) were invited to participate the survey by either completing the Web-based questionnaires, or by scheduling a face-to-face interview with the authors. By the end of April (roughly 3 weeks) after the survey was completed, the authors received 18 responses on the Web-based questionnaires and had 2 face-to-face interview, for a response rate of ~45%.

Additionally, the survey managed to collect data from three continents (Europe, North and South America) and from representants of several major application lifecycle phases. Based on a thorough analysis of the responses, we drew the following conclusions:

• Application lifecycle management is relevant and important for all interviewed (ranging from very small to very large) organization and all interviewed roles.

• Over 60% of all organizations already do lifecycle management in some form (either applying a particular ALM solution or by integrating different tools, in order to achieve the same results). These homemade integrations are only connected to some subset, so you may have better requirements but it stops short of helping the developers.

• Most interviewed organizations rely on multiple tools from several different vendors.

• Meeting the needs of the business is the top priority, therefore switching to new tools is often unfeasible (due to missing migration strategies between the tools, missing know-how within the organization, productivity losses, etc.)

• The majority (over 75%) of interviewed find the current forms of tool integration (through an ALM system or through an system buses) very difficult to implement and find the final results unsatisfactory. Some of the most common problems are semantic loss, when moving from one tool to another, redundant data, automation limitations, no round-trip support

• Mapping of interdependencies and inter-relationships at the elemental, process, and services level is unsatisfactory

• No single ALM tool vendor can build everything he needs to build to support very complex software development lifecycles and software development supply chains, as every organization, even every project is doing something slightly different than all the others - they have investments in various tools, they are using a range of processes, and they are deploying to a wide variety of platforms and technologies The range of necessary tools, which have to be integrated varies from project to project and from organization to organization, therefore an optimal ALM tool needs to be able to include, exclude different tools as needed.

• Even if such a vendor succeeds to design such a "everything to everyone" tool, the majority of interviewed would still prefer a deep semantic integration of their existing tools into an ALM system, rather than taking the liability of a single vendor solution.

The reminder of this analysis concentrated on deriving concrete use cases and features, which define how the ALM solution might be utilized, and non-functional requirements defining non-functional qualities of the system. These use cases, features (represented as abstract use cases) and non-functional requirements, formed the basis of the UML’s use case and requirements models, an export from which is shown in Appendix E.
The use case model is structured in packages according to the different phases in the application lifecycle process and/or their respective sub-phases, and each package contains a diagram with the specific use cases relevant for this phase/sub-phase. Additionally, in order to avoid redundancy, the authors defined a separate package containing use cases shared between the different other packages. These use cases are then included in the respective diagrams as needed, without having to duplicate them. The use case model also included the roles (as identified in the stakeholder analysis) that interact with the ALM system. Their activities and interactions are described by the flow of events in the use cases, which describes what an actor does and how the ALM system reacts, but not how the system functionality is realized. The following graphic represents an arbitrary example of such a Use Case Diagram and an excerpt of the Use Case model structure.

![Figure 17: An example of a Use Case Diagram](image1)

![Figure 18: Excerpt of the Use Case Model structure](image2)
A second top-level package describes the non-functional requirements, which describe the quality attributes of the system as it performs its tasks and therefore impose constrain on the design and implementations. These are structured according to the IEEE-Standard 830 from 1993, which identifies thirteen possible categories of non-functional requirements. The following graphics represent an arbitrary example of a requirements diagram and an excerpt of the requirements structure.

![Example of a requirements diagram](image1.png)

![Excerpt of the Requirements Model structure](image2.png)
5.3 Identification of improvement areas based on a delta analysis of current ALM solutions and the elicited requirements for a future ALM approach

The following section briefly introduces some of the most popular ALM tools currently on the market and summarizes their advantages and disadvantages based on the insights gained from the analysis of the survey results. Afterwards the authors identify the major shortcoming and improvement areas of these tools (representative for all available ALM solutions) evaluated against the elicited user requirements. These shortcomings are later (in the following section) addressed with the improvement design concepts and ideas.

5.3.1 Microsoft Visual Studio with Team Foundation Server

We will start our analysis with Microsoft’s Visual Studio with Team Foundation Server (TFS2010), which appears to be the most comprehensive ALM implementation currently available. The main reason for this is that unlike other competitors' ALM solutions, which are often based on the companies different software tools stitched together with fragile bridges and connectors, TFS2010 was designed from ground up as an ALM system and provides the majority of the functionality required to successfully support the application lifecycle management process.

Roughly summarized TFS2010 is a shared repository for all kinds of development artifacts, which allows to automate development tasks beyond the scope of a single SDLC phase and support the collaboration and communication of the different roles involved in a products development. TFS2010 is based on the concept of work items, and referring to units of work, which needs to be executed or completed, and can represent several different types of information (e.g. bug, problem, requirement, script, task, risk, etc.). Depending on the currently chosen TFS2010 framework different kinds of information are available for any given project and these types of information have different attributes. These work items can be connected with each other, in such a way providing traceability among all the different supported information artifacts (Vinay, 2011; Wikipedia, 2011j).

Architecturally TFS2010 is based on a standard three-layer architecture: client, application, and data tier working along with SQL server database to provide a central store for information, as shown on the following graphic(Informer Technologies Inc, 2011):
In addition to advanced version control, the TFS2010 suite offers integration with several other Microsoft products (e.g. Sharepoint Services, MS Project, etc.) to extend its build-in capabilities like work-item tracking, source code control, automated builds, metrics and reporting, project portals, document sharing and team collaboration with additional functionalities provided by other tools. The full range of functionality provided by TFS2010 can be seen in the following graphic (IBM Inc., 2010):

As a pure server solution TFS2010 does not provide a user, but rather it exposes web services which client applications, like Microsoft’s own Visual Studio 2010 or the popular Eclipse IDE, can use to integrate TFS functionality with themselves. The integration with these client applications, in particular with MS Visual Studio,
provides further relevant functionalities for Architects, Developers and Testers. In addition TFS2010 provides a rich web interface, which can be used for remote access (UpgradeCompetenceCentre, 2008).

One of the most interesting features provided by TFS2010 is the so-called process templates. These define the process model applied in the project and in such a way the types of work items available through the project, their respective state transitions and traces among each other. These allows for the definition of rules to enforce the adherence to the chosen process model as well as the automation of recurring tasks (e.g. nightly builds) (UpgradeCompetenceCentre, 2008). The following table summarizes how the different TFS2010 functionalities map to the different ALM process areas (Patchotepong & Koifman, 2009):

<table>
<thead>
<tr>
<th>ALM Process Area</th>
<th>TFS2010 Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Interface Design</td>
<td>Using Architecture features in VSTS</td>
</tr>
<tr>
<td></td>
<td>Project Portals</td>
</tr>
<tr>
<td>Project Management</td>
<td>Metrics and Reporting (reporting based on reporting services)</td>
</tr>
<tr>
<td></td>
<td>Office Project Integration</td>
</tr>
<tr>
<td>Requirements Engineering</td>
<td>Office Excel Integration</td>
</tr>
<tr>
<td></td>
<td>TFS Template (MSF for CMMI or MSF for Agile) with work item tracking</td>
</tr>
<tr>
<td></td>
<td>Project Portals</td>
</tr>
<tr>
<td>Quality &amp; Test</td>
<td>Automated Test and Quality Metrics (features in VSTS)</td>
</tr>
<tr>
<td></td>
<td>Automated Build</td>
</tr>
<tr>
<td></td>
<td>Project Portals</td>
</tr>
<tr>
<td>Code &amp; Code Quality</td>
<td>Development Features in VSTS</td>
</tr>
<tr>
<td></td>
<td>Quality Metrics &amp; Quality Analysis</td>
</tr>
<tr>
<td></td>
<td>Check-in Policy</td>
</tr>
<tr>
<td>Architecture &amp; Design</td>
<td>Using Architecture and Modeling features in VSTS</td>
</tr>
<tr>
<td>Software Configuration Management</td>
<td>Source code control</td>
</tr>
<tr>
<td>Product, Process, Portfolio and Variability Management</td>
<td>Portfolio Management</td>
</tr>
<tr>
<td></td>
<td>Stakeholder Communication</td>
</tr>
<tr>
<td>IT Services &amp; Operation</td>
<td>Infrastructure Architecture</td>
</tr>
<tr>
<td></td>
<td>Infrastructure Deployment</td>
</tr>
</tbody>
</table>

Table 1: Mapping between the ALM process areas and the functionality provided by TFS 2010 (Schäfer & Melich, 2009)

As part of a later validation process, described in chapter 6, the authors together with the validation group defined a list of important characteristics of ALM tools, which will be briefly examined here in the context of TFS 2010. The examination will be based on the “traffic light principle” (green, amber, red), where:

<table>
<thead>
<tr>
<th>Green</th>
<th>Fully Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amber</td>
<td>Partially Supported</td>
</tr>
<tr>
<td>Red</td>
<td>Not Supported</td>
</tr>
</tbody>
</table>

Table 2: Characteristic support status

In regards to these characteristics the evaluation in the context of TFS 2010 produces the following results:
### Characteristic

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Support Status</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Interoperability with other tools</td>
<td></td>
<td>Good interoperability with other MS tools, however support for third parity tools requires the development of point-to-point integrations.</td>
</tr>
<tr>
<td>2. Support for all ALM process areas and perspectives</td>
<td></td>
<td>Due to the “missing” interoperability with third parity tools, TFS can only difficulty be extended beyond its out-of-the-box support for the ALM process, which is focused on the SDLC phases of ALM.</td>
</tr>
<tr>
<td>3. Traceability</td>
<td></td>
<td>Limited to the out-of-the-box supported ALM process phases. Traceability to external tools required the development of synchronization mechanisms.</td>
</tr>
<tr>
<td>4. Reporting</td>
<td></td>
<td>Through project dashboards, however reporting is also limited to the supported out-of-the-box process areas.</td>
</tr>
<tr>
<td>5. Automation</td>
<td></td>
<td>Automation if available only for the areas supported out-of-the-box. If third parity tools are used alongside TFS 2010, this limits the possibility for an end-to-end automation.</td>
</tr>
<tr>
<td>6. Extendibility / Scriptability</td>
<td></td>
<td>SDK is available.</td>
</tr>
<tr>
<td>7. Scalability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8. Distributed working</td>
<td></td>
<td>Access through a web browser.</td>
</tr>
<tr>
<td>9. Automatic Data Migration from other tools</td>
<td></td>
<td>Not available.</td>
</tr>
<tr>
<td>10. External factors (Price, Loss of Know How, etc.)</td>
<td></td>
<td>High cost of migration and loss of know how, due to a migration away of known tools. Also staff will require training and time to reach the same productivity level.</td>
</tr>
</tbody>
</table>

### Summary

While TFS 2010 is a very comprehensive solution, which covers nearly all stages in the project life-time and provides deep integration with developer tools and services for successful organization of software development processes and real-time control, in the author’s opinion there also are many negative sides to it. As a monolithic solution provided by a sole supplier, like in this case Microsoft, it does not support heterogeneity as the new norm for a modern ALM tool stack and in such a way limiting diversity and innovation. Instead it depends on the very particular application lifecycle management (ALM) stack, implemented by Microsoft, which in this specific case means that TFS can be integrated “only with Microsoft-made messaging and only with the use of a power tool” (Patchotepong & Koifman, 2009). That means that in order to introduce new tool, offering a new or missing functionality into the TFS system (e.g. Variability Management) or to replace one of TFS’s build-in components with an external one, one needs to implement a fragile bridge or a connector transforming the particular tools internal object model to TFS’s object model, in such a way ultimately leading to a situation, which the initial introduction of an ALM system tried to improve. This, enforced “single mindedness”, also means that, on the one side,
before TFS can be introduced into an organization, all previously used databases (e.g., RMDB, Test Management DB, etc.) need to be migrated to TFS 2010—a process, which can be both expensive and time consuming, and on the other side all acquired experience and know-how with the old tools is irreversibly lost. Last, but not least, TFS 2010 can be deployed only on Windows platforms, which greatly limits the choices the customers have, and can even be a “deal-breaker” for some smaller companies (Schäfer & Melich, 2009).

5.3.2 SAP Solution Manager

SAP Solution Manager is a software product of German SAP AG Company, and a platform that helps to improve the efficiency of the ALM in successful integration and collaboration with other tools. SAP covers several ALM activities. The platform of SAP Solution Manager is a crucial part, providing the ALM functionality. SAP Solution Manager allows for all the essential information, tasks, tools and methods to be accessed. The ALM operations are made faster and at lower cost with the use of SAP Solution manager throughout the whole lifecycle of a product. SAP Solution Manager is used to manage the various SAP systems. The following figure represents how SAP Solution Manager is used throughout the lifecycle of the application in accordance to ALM (Patchotepong & Koifman, 2009).

![Figure 23: SAP Solution Manager covering ALM (Patchotepong & Koifman, 2009)](image)

The first phase, called the blueprint, involves the choosing of the project, for which SAP Solution Manager is equipped. The scenario is either selected from repository, or is made by team. After the project is configured, according to (BMC-Software, 2009), SAP Solution Manager provides services for the managing and implementation of the configuration. The next stages are the testing and further development that takes place with the assistance of SAP Solution Manager.

The implementation of the project involves the following steps, according to (BMC Software, 2006):

- “Continuous monitoring of project targets, especially costs, deadlines, and resources
Active implementation of Organizational and Change Management
Elaboration of training materials and end user documentation
Creation of end user roles and implementation of authorization and security concept
Elaboration and implementation of realization specifications for customer developments’’

The customers of SAP AG are offered the business support of implementation as well as the technical one. In order to achieve better results “e-learning module is an integrative part of the Solution manager and designed to support the training activities of the project” (Rapid Technologies, 2010).

Schafer et al state that SAP Solution Manager assists its customers “in configuring your systems around your processes” and that “by integrating existing testing tools, SAP Solution Manager supports …[customers] in organizing and documenting the results of a range of testing activities, from unit testing through to integration testing” (Rapid Technologies, 2010).
Table 4 shows the evaluation results based on the criteria (characteristics), defined during the validation session.

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Support</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Interoperability with other tools</td>
<td>Not very good</td>
<td>Not very good interoperability, and one of the weak sides of SAP Solution Manager, because of the difficult integration of other tools.</td>
</tr>
<tr>
<td>2. Support for all ALM process areas</td>
<td>SAP Solution</td>
<td>SAP Solution Manager offers support for all of the phases of ALM, with the exception of other tool integrations.</td>
</tr>
<tr>
<td>and perspectives</td>
<td>Manager</td>
<td></td>
</tr>
<tr>
<td>3. Traceability</td>
<td>SAP has very</td>
<td>SAP has very powerful traceability functions that covers all information areas (business processes, maintenance activities, etc.), but it is not possible to trace the third-party tools.</td>
</tr>
<tr>
<td></td>
<td>powerful</td>
<td></td>
</tr>
<tr>
<td></td>
<td>traceability</td>
<td></td>
</tr>
<tr>
<td>4. Reporting</td>
<td>Through extended changed analytics</td>
<td></td>
</tr>
<tr>
<td>5. Automation</td>
<td>Automation</td>
<td>Automation offers end-to-end functionality out-of-the-box or separately made by a customer.</td>
</tr>
<tr>
<td></td>
<td>offers</td>
<td></td>
</tr>
<tr>
<td></td>
<td>end-to-end</td>
<td></td>
</tr>
<tr>
<td>6. Extendibility / Scriptability</td>
<td>SAP Business</td>
<td>SAP Business One SDK</td>
</tr>
<tr>
<td></td>
<td>One SDK</td>
<td></td>
</tr>
<tr>
<td>7. Scalability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8. Distributed working</td>
<td>Access through</td>
<td>Access through a web browser.</td>
</tr>
<tr>
<td></td>
<td>a web browser.</td>
<td></td>
</tr>
<tr>
<td>9. Automatic Data Migration from other</td>
<td>Not available.</td>
<td></td>
</tr>
<tr>
<td>tools</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10. External factors (Price, Loss of</td>
<td>High licence</td>
<td>High licence cost; the integration interface usability is bad.</td>
</tr>
<tr>
<td>Know How, etc.)</td>
<td>cost</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: SAP Solution Manager Evaluation results

Summary
The drawback of the SAP Solution Manager is that it requires the employee to be trained in order to use it. Managing SAP related problems requires skills and experience as well. For those customers who “want to have a central ticketing system to cover everything in the organization, […] there is a license cost and integration interface to consider for this extension” when it comes to SAP Solution Manager (BMC Software, 2006). Thus having SAP Solution Manager for a platform is a tough
solution and Patchotepong and Koifman (Borland, 2011a) recommend for the customer to view other platform offer for comparison when they are planning to pursue the most benefit out of the organization.

5.3.3 BMC Remedy IT Service Management Suite

BMC Suite covers the following management processes’ services:

- BMC Remedy Asset Configuration Management
- BMC Remedy Change and Release Management
- BMC Remedy Service Desk
- BMC Service Level Management

According to BMC Software, the BMC Suite enables “ITIL aligned service management processes for the IT department”, and “provides seamless, out-of-the box workflow automation within and across key ITIL processes resulting in rapid organizational adoption and process efficiency” (Borland, 2011a).

BMC Remedy Asset Configuration Management application allows the user to “track and manage enterprise configuration items […] and their changing relationships throughout the entire [configuration] life cycle” (Borland, 2011b).

The BMC Remedy Change and Release Management deals with the newly required change and the endorsement of that change. When changes are to be made, the assessment of risks and consequences of the change are researched under different scenarios. The approval of the change is made in accordance with as many factors as possible. The release stage happens after considering all the solutions, and choosing the best solution for that change. The BMC Change and Release Management is one of the most important features of the whole suite, as it deals with the crucial decision that will influence or alter other different business decisions/processes within the organization. The figure below demonstrates the BMC Change and Release Management in greater detail (Borland, 2007).

![Figure 24: BMC Remedy Change and Release Management (Borland, 2007)](image)

BMC Remedy Service Desk includes two different ways of managing. One is that of managing incidents, in other words, when user makes an enquiry, and the response must be made to such an alert, call, etc. The other is of investigating and managing the
problems, involving such practices as fixing of the bugs, correcting errors, adjusting the database, etc.

BMC Service Level Management enables the user to “formally document the needs of its customers or lines of business using service level agreements, and provide the correct level of service to meet those needs” (Borland, 2007).

Table 5 shows the evaluation results for the BCM Remedy IT Service Management Suite against the predefined evaluation characteristics.

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Support Status</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Interoperability with other tools</td>
<td>Interoperability is not very good, and it is difficult between version of BMC, and BMC does not have many third party tools integrations.</td>
<td></td>
</tr>
<tr>
<td>2. Support for all ALM process areas and perspectives</td>
<td>BMC Remedy focuses on ITIL service management processes.</td>
<td></td>
</tr>
<tr>
<td>3. Traceability</td>
<td>Traceability of changes is approved in BMC for the internal processes.</td>
<td></td>
</tr>
<tr>
<td>4. Reporting</td>
<td>BMC offers ROI metrics reporting.</td>
<td></td>
</tr>
<tr>
<td>5. Automation</td>
<td>BMC Remedy offers out-of-the box workflow automation for the main ITIL processes.</td>
<td></td>
</tr>
<tr>
<td>6. Extendibility / Scriptability</td>
<td>SDK is available.</td>
<td></td>
</tr>
<tr>
<td>7. Scalability</td>
<td>BMC Remedy AR system.</td>
<td></td>
</tr>
<tr>
<td>8. Distributed working</td>
<td>Distributed server option.</td>
<td></td>
</tr>
<tr>
<td>9. Automatic Data Migration from other tools</td>
<td>Not available.</td>
<td></td>
</tr>
<tr>
<td>10. External factors (Price, Loss of Know How, etc.)</td>
<td>Not popular, loss of know how.</td>
<td></td>
</tr>
</tbody>
</table>

Table 5: BCM Remedy Evaluation results

Summary

The problem with BMC Remedy suite is that without custom work some of the integrating cannot be accomplished. Custom coding is necessary for some typical functions, such as an import of data from one system to another, etc. BMC Remedy is not very popular, and does not have many third party tools integrations. It has also been so for many years, so the problem is not new. As a consequence, the configuration and development stages are more difficult to accomplish.

5.3.4 Borland StarTeam & the Open ALM concept

With its “Star Team” solution, Borland combines version control with change management and tries to “improve team communication and collaboration through centralized control of a project’s digital assets and the activities which drive change to those assets” (Borland, 2007). While it may not excel in either area, by linking revisions and workflow based change management, StarTeam allows managing assets and activities in a single repository without having to integrate or synchronize multiple repositories or tools, which is the tool's greatest strength. Additionally StarTeam offers a comprehensive range of abilities, including integrated change management, defect tracking, file versioning, requirements management, threaded discussions and version control, as well as project and task management capabilities, which provide the functionality, security, and scalability to meet the software configuration and change management requirements of many organizations (Riley, 2007). In order to achieve
this, in 2007 Borland announced the introduction of a new ALM framework, called “open ALM”, which promises to allow the use of any lifecycle tool, or platform that fits the user’s particular needs, rather than having to compromise to fit a specific tool vendor, in such a way accounting for enormous cost saving (in terms of licenses, know how and time) by utilizing and incorporating already existing tool investments, rather than introducing a completely new one, while still allowing to manage the application life cycle, to make it measurable, predictable and improvable even across the different lifecycle sub-processes (Riley, 2007). The following graphic shows the Open ALM architecture, as envisioned by Borland (Göthe et al., 2008):

![Figure 25: The OpenALM architecture (Göthe et al., 2008)](image)

According to (IBM, 2011a; 2011b) with the open ALM Borland envisions the following characteristics for the future of ALM:

- Support for any process
- Support for any tool and/or tool combination, regardless of vendor
- Support for all major platform, in order to enable organizations to deploy their application products to a broad range of products.
- Automation of third-party data collection to allow cross measurement and automation.

In order to achieve this, the open ALM platform includes two important components, which are shared across all solution areas (Avnet Technology Solutions, 2011):

- **ALM meta-model** - providing a unified information model, aimed at defining the elements and relationships, necessary to describe all processes part of the application life cycle and which all other tools will have to implement.
• **ALM integration layer** – providing and unified and common way, in which the ALM tools can be embedded and integrated into the ALM platform by implementing a conform custom adapter, which will connect them to the platform and allow the automatic pulling of data, harvesting of metrics and traceability.

Following the StarTeam & Open ALM evaluation, based on the identified characteristics:

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Support Status</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Interoperability with other tools</td>
<td></td>
<td>The idea of Open ALM is to support the easy integration of third parity Tools, however the vendor never achieved this goal, as it could not motivate other tool vendors to adopt the unified ALM meta-model and to support the ALM integration layer.</td>
</tr>
<tr>
<td>2. Support for all ALM process areas and perspectives</td>
<td>Supports out of the box the so-called “core process areas”. The idea is that all other missing functions could be provided by integrating third parity tools using the Open ALM framework, which currently lacks adoption (see 1.)</td>
<td></td>
</tr>
<tr>
<td>3. Traceability</td>
<td>Support for end-to-end traceability for all tools integrated through the Open ALM framework. Unfortunately due to the limited adoption of the framework, the out-of-the box supported “core process areas” limit traceability.</td>
<td></td>
</tr>
<tr>
<td>4. Reporting</td>
<td>Through project dashboards, automatic alerts, etc., however the same problem of missing adoption applies here.</td>
<td></td>
</tr>
<tr>
<td>5. Automation</td>
<td>Automation if possible, however the same problem of missing adoption applies here.</td>
<td></td>
</tr>
<tr>
<td>6. Extendibility / Scriptability</td>
<td>SDK is available.</td>
<td></td>
</tr>
<tr>
<td>7. Scalability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8. Distributed working</td>
<td>Not necessary.</td>
<td></td>
</tr>
<tr>
<td>9. Automatic Data Migration from other tools</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10. External factors (Price, Loss of Know How, etc.)</td>
<td>Low cost of migration and no loss of know how, as open ALM does not require a migration away from the currently used tools.</td>
<td></td>
</tr>
</tbody>
</table>

Table 6: Open ALM Evaluation results

**Summary**

Despite the great idea and all good and worthwhile principles of Borland, when launching the Open ALM initiative back, which if successful would have alleviated a problem, that has “plagued the expensive and highly proprietary ALM market since its inception” (Avnet Technology Solutions, 2011), namely the absence of a standard means of integrating all the “moving components” in the software development process, the project was abandoned shortly after it’s initial announcement, due to a
5.3.5 Rational Collaborative ALM (C/ALM) & IBM’ Jazz Integration Architecture

IBM Collaborative ALM (C/ALM) is another ALM solution, which allows teams to collaborate and coordinate development life cycle activities, including requirements, modeling, development, build and testing, using integrated iteration planning, work item management, source control, builds management, end-to-end traceability, visibility, automation and reporting capabilities, based on the Jazz platform. To define C/ALM IBM has produced, a so called ALM blueprint, illustrated in the following picture, and defining three different layers (SDLC Blog, 2009):

- A Project Health Dashboard to allow the active and continuous management of the project and to help monitor the project at any point.
- A Layer for the discipline-specific modules, like requirements engineering, modeling, build management, etc. with automation and scanning underpinning all of them.
- Finally a foundation level, which helps to integrate all of the different tools, so that the activities they perform and the assets they produce contribute to the delivery of a software application in an efficient and streamlined manner.

![C/ALM blueprint](SDLC Blog, 2009)

IBM’s C/ALM is based around the Jazz platform, whose key challenge is achieving integration, while preserving the loose coupling between the different tools. This makes the approach of the Jazz platform a game-changer, because it allows creating a single consistent platform, which allow to integrate not only the different IBM solutions, but also to develop own custom solutions and is open to third-parity developers as well. In such a way Jazz tries to break down the walls between the different specialized tools involved in the application life-cycle, allowing team-members to easily share data and information in the context of their work. To achieve this Jazz federates data across independent databases using different Internet protocols and can integrate data from where it resides without having to import/export it between the different tools. This is done by defining a set of so called foundation services, which can be adopted by other tools selectively to implement cross-tool capabilities, or by exposing tool-specific services with a language-neutral, RESTful interface so other clients can access them (Stafford, 2010).
Additionally IBM Rational, provides several ready and jazz-able solutions, which can be tightly integrated to construct a Rational Collaborative ALM as shown on the following figure (McCarthy, 2011).

![Rational C/ALM](image)

**Picture 27: Rational C/ALM (McCarthy, 2011)**

The key components of such a solution are (HP, 2011; Wikipedia, 2011k):

- **Rational Team Concert (RTC)**, which supports the process control and customization and offers several out-of-the box process models, including Scrum and Agile methods, and they all can be additionally tailored to meet the specific needs of the adopting organization.
- **Rational Requirements Composer (RRC)**, which supports the process of requirements definition and management and offers capabilities like graphical modeling, storyboarding, support for review workflows, etc.
- **Rational Quality Manager (RQM)**, which supports the quality assurance process from both technical, as well as business perspective by addressing the needs of business analysts and QA professionals and providing the ability to use different perspectives for accessing and viewing testing assets, based on the user's role.
- **Rational Insight**, which is a performance measurement and management solution, which helps to improve project and process performance.
- **Rational Asset Manager**, which helps to understand what assets are available in an organization, how they are related, and if they are delivering business value.

Following also the IBM Jazz platform evaluation, based on the identified characteristics:

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Support Status</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Interoperability with other tools</td>
<td></td>
<td>The idea of IMB’s Jazz platform is to support the easy integration of third parity through RESTful Service integration, however until now this goal could not be fully achieved, as IBM could not motivate enough tool vendors to make their tools “Jazz-able”.</td>
</tr>
<tr>
<td>2. Support for all ALM process areas and perspectives</td>
<td></td>
<td>The Rational Team Suite support “out-of-the-box” a large part of the ALM process. The idea is that all other missing functions could be provided by using third parity tools which integrate with the underlying Jazz platform, which currently lacks adoption (see 1.). Not even all Rational</td>
</tr>
<tr>
<td>3. Traceability</td>
<td>Support for end-to-end traceability for all tools integrated through the Jazz platform. Unfortunately due to the limited adoption of the framework, the traceability is also limited only to the currently supported tools.</td>
<td></td>
</tr>
<tr>
<td>4. Reporting</td>
<td>Through project dashboards, automatic alerts, etc., however the same problem of missing adoption applies here.</td>
<td></td>
</tr>
<tr>
<td>5. Automation</td>
<td>Automation if possible, however the same problem of missing adoption applies here.</td>
<td></td>
</tr>
<tr>
<td>6. Extendibility / Scriptability</td>
<td>SDK is available.</td>
<td></td>
</tr>
<tr>
<td>7. Scalability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8. Distributed working</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9. Automatic Data Migration from other tools</td>
<td>Not necessary.</td>
<td></td>
</tr>
<tr>
<td>10. External factors (Price, Loss of Know How, etc.)</td>
<td>Low cost of migration and no loss of know how, as open ALM does not require a migration away from the currently used tools.</td>
<td></td>
</tr>
</tbody>
</table>

Table 7: Open ALM Evaluation results

Summary
Although IBM’s ALM vision match up what the authors are trying to achieve and when looking at the jazz.net site, one can see a trend, that an increasing number of IBM tools are adopting the Jazz platform, for it to be really successful and to achieve what it promises, it needs to be accepted also by third parity tool vendors (HP, 2011; Wikipedia, 2011k). Currently some of the most widespread tools on the market do not provide support for the Jazz platform, including some of IBM’s own solutions (e.g. Rational Doors, etc.), which makes the vision of creating an ALM solution based on a free choice of best-of-breed tools and a common platform and meta-model still not possible. Although there are several IBM business partners contributing to extend IBM’s Jazz based products with some new features, but even they do not seem willing to adopt the Jazz platform for their own products. This demonstrates that it is not easy for a major ALM vendor to get the other major ALM vendors to participate. Partially this might be due to distrust, partially it could be the attempt to protect the monolithic stacks of tools from best-of-breed purchasers, but in the author’s opinion it is due to the way IBM is trying to attract third-parity vendors and external developers - namely by trying creating a community around the jazz.net and OSLC websites, which requires from the other ALM vendors considerable investment and offers them little benefits, rather than open sourcing the project, as they so successfully did with the eclipse platform (SDLC Blog, 2009).

5.3.6 HP ALM 11

HP ALM 11 is a role-based system aimed at automating and integrating the diverse HP tools, which handles application processes from development through requirements management and quality and performance testing through deployment (Rich, 2010). In such a way it provides a support for all the different roles and stakeholder involved in the application life-cycle to better “automate the ability to tie line-of-business requirements directly in with the workflow of IT software professionals who design, spec out, develop and test” (Rodriguez, 2008).

HP ALM 11 offers the following ALM features (Lott, 2008):
• Project planning and project tracking support to provide real-time visibility into the project and to help manage its progress.
• 3-Way Traceability support to allow stakeholders to trace between requirements, development and quality artifacts.
• Support for different methodologies, like waterfall, agile, hybrid, etc., which can be further customized to fit the particular organization.

The following picture shows the different components of HP ALM 11 (Burmester et al., 2004):

![HP Application Lifecycle Management](image)

Figure 28: HP ALM 11 (S. L. Wright, 1994)

• **HP Business Process Testing** to support the organizations quality assurance processes by allowing to easily designing complex software test scenarios utilizing reusable test components.
• **HP Functional Testing** is another quality assurance tool used to simplify the automation of functional and regression testing.
• **HP Load Runner** is a tool used for automated performance and load testing and for examining system behavior and performance, while generating actual system load.
• **HP Performance Testing** is another performance testing platform and framework used to standardize, centralize and conduct performance testing.
• **HP Quality Center** is a quality management platform used for the management of tests and other quality relevant artifacts across the entire application lifecycle. Quality Center additionally provides modules to support the requirements management, release and cycle management, test management, defect management.
• **HP Service Test** is a SOA testing solution that simplifies and accelerates the automated functional testing of SOA services.
• **HP Sprinter** is a new tool to support the manual testing process.
• **HP Test Data Management** is structured, automated test data management tool that employs automated data sub-setting and masking.
• **HP Unified Functional Testing** is an automated functional defect testing solution with support for both GUI and non-GUI testing.

Following the HP ALM 11 evaluation, based on the identified characteristics:

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Support</th>
<th>Remark</th>
</tr>
</thead>
</table>
| HP Application Lifecycle Management
| A single platform for managing the application lifecycle
| ![HP ALM Platform](image) |         |        |

[54]
<table>
<thead>
<tr>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Interoperability with other tools</td>
</tr>
<tr>
<td>Interoperability is rather difficult and only</td>
</tr>
<tr>
<td>possible to developing point-to-point</td>
</tr>
<tr>
<td>integrations with other tools.</td>
</tr>
<tr>
<td>2. Support for all ALM process areas and</td>
</tr>
<tr>
<td>perspectives</td>
</tr>
<tr>
<td>Currently limited to test areas and with</td>
</tr>
<tr>
<td>some rudimentary support for</td>
</tr>
<tr>
<td>requirements engineering.</td>
</tr>
<tr>
<td>3. Traceability</td>
</tr>
<tr>
<td>Offers traceability only between test and</td>
</tr>
<tr>
<td>requirements engineering artifacts.</td>
</tr>
<tr>
<td>4. Reporting</td>
</tr>
<tr>
<td>Through project dashboards, automatic</td>
</tr>
<tr>
<td>alerts, etc., however the same problem of</td>
</tr>
<tr>
<td>missing adoption applies here.</td>
</tr>
<tr>
<td>5. Automation</td>
</tr>
<tr>
<td>Automation if possible, however limited</td>
</tr>
<tr>
<td>to the supported ALM process areas.</td>
</tr>
<tr>
<td>6. Extendibility / Scriptability</td>
</tr>
<tr>
<td>SDK is available.</td>
</tr>
<tr>
<td>7. Scalability</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>8. Distributed working</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>9. Automatic Data Migration from</td>
</tr>
<tr>
<td>other tools</td>
</tr>
<tr>
<td>Not available.</td>
</tr>
<tr>
<td>10. External factors (Price, Loss of</td>
</tr>
<tr>
<td>Know How, etc.)</td>
</tr>
<tr>
<td>High cost of migration and loss of know</td>
</tr>
<tr>
<td>how, due to a migration away of known tools.</td>
</tr>
<tr>
<td>Also staff will require training and</td>
</tr>
<tr>
<td>time to reach the same productivity level.</td>
</tr>
</tbody>
</table>

Table 8: Open ALM Evaluation results

Summary

While HP ALM 11 provides organizations with a centralized application management platform for managing, measuring and automating applications throughout the complete development life cycle, supporting an connecting all core lifecycle activities from a workflow perspective, it similarly to Microsoft’s TFS, depends on the very particular application lifecycle management (ALM) stack, and can be integrated mainly with other HP tools in such a way limiting the heterogeneity and introducing new costs for licenses, migration strategies, etc. to adopting organizations. Additionally no single vendor, neither Microsoft nor HP, can build everything he needs to build to support very complex software development lifecycles and software development supply chains, which is why every organization, even every project is doing something slightly different than all the other (combine different products, or build they home-grown products, etc). Not being open to such heterogeneity is often a no-go for many companies.

5.3.7 Summary

As the tool analysis, which does not demand to be complete, shows there are several ALM tool options out there, which, present viable options for interested organizations. In the authors opinion, however all these tools share a few element, which prevents them from being the industrial ALM solution, envisioned by the authors and requested by the interviewed stakeholders:

- The currently available ALM solutions are largely concentrated on the software development aspect of the application life cycle (SDLC) and often have no out-of-the box support for the other aspects of ALM (APM and Service Management and Operations). Therefor their true value (in the sense of a full-features ALM solution) can only be achieved by integrating them with tools that cover the other aspects of ALM. Even if there were an ALM solution offering support for the entire application lifecycle, the interviews have shown that every project adheres to a slightly different
process and requires its own set of tools, which are often not required for other tools. A tool offering everything possible is not feasible, as it would imply a very high price and high complexity due to the much functionality necessary to be managed. Therefore being able to integrate arbitrary tools when needed is crucial for the industrial ALM solution expected by the stakeholders.

- Most of the currently available ALM solutions are monolithic and implement a single particular ALM stack, and because no two ALM stacks look alike the integration of these tools with other tools is rather difficult and only possible through fragile bridges and connectors implemented using the often provided software developer kits. Such bridges and connectors, however go against the entire concept of integrated ALM solution. Additionally the issue of data migration from the old to the new tools, as well as the missing knows how and experience of the employees makes such solutions very costly and time consuming and often unfeasible for operating companies, where meeting the needs of the business is the top priority and often implies unwillingness to temper around with suboptimal but working system.

- Even the few available efforts aimed at creating a heterogeneous and/or open ALM solution (e.g. Borland and/or IBM), allowing the easy integration of best-of-breed tools are currently unsuccessful, due to the missing commitment of the other ALM vendors to follow along. This missing commitment is due to one of two reasons (Reiss, 1990; 1999; Yap, Chiong, Grundy, & Berrigan, 2005):
  - A distrust and fear to open up to other large ALM vendors and to contribute to and invest in a project set-up by a competitor, which initially offers them little benefits.
  - An attempt to protect the monolithic stacks of tools from best-of-breed purchasers in an effort to save their market share.

5.4 Design concepts and ideas

The following section briefly presents the ideas for a new improved design of an integrated and industrial ALM solution, which were generated as part of a brainstorming session, and are intended as an initial reference point for setting up a new research agenda aimed at solving the abovementioned problems. The ideas are, as typical for a brainstorming session only recorded (see Appendix F) and briefly explained here, but not “judged”. They need to be evaluated and “criticized” from the to be set-up task force as a part of the to be set-up project aimed to press ahead with the goals and objectives laid out by our work.

- Integration Means
  - RESTful Web services – During their work on the Jazz platform, the IBM Jazz team decided that integrations, rather than implementations have to be the focus of Jazz enablement. Therefore the team started to look for reference architectures, which enable such massive scaling, extensibility and provide a rich model for linked data. They came to realize that the Web could provide good reference architecture, if the team manages to implement fundamental Web capabilities for the data in the different tools. In particular such capabilities should include stable URLs for important resources, should enable the publishing of resource formats, and maintain compatibility, should enable the publishing of specifications for common services and enable OAuth for authorizing tool-to-tool communications (Emmerich, Arlow, Madec, & Phoenix, 1997; THOMPSON, 1992). The RESTful API is a flexible way to expose system’s resources and in such a way to provide different application with standardized data. In such a way it can help to meet the integration requirements for building systems where data can be easily combined, as it is the case with the envisioned ALM 2.0 (I. Thomas & B. Nejmeh, 2002). One of the major issues with designing RESTful services it be find a way to make
all relevant application resources available through URI’s and this as cleanly as possible (Wikipedia, 2011).

- **Metamodel-level integration** - Most practitioner tools today have their own data format and data metamodel, which complicates their integration. Achieving consistency between the metamodels used by the different practitioner tools by either defining a end-to-end metamodel describing the entire application lifecycle ecosystem, including all entities, process relationships (traceability) and measurement units (metrics) as a standard for all tools participating in the application lifecycle, or by defining rules and patterns, which would enable to automatically integrate different meta-models, would greatly simplify the integration between different tools (A. Brown, Johnston, Larsen, & Palistrant, 2005).

- **Domain-Specific Language (DSL) based Integration** – DSLs can be used in combination with the previously described meta-model integration to define and describe the rules and patterns between the different meta-models and in such a way to facilitate their automatic integration.

- **Multi-level orthogonal integration** - the currently prevalent point-to-point integrations between different practitioner tools, do not allow for achieving the higher level of software integrity necessary for the kind of ALM solution, the author have envisioned. In order to achieve such deep integration, the different tools need to be connected on several orthogonal levels (S. L. Wright, 1994):
  - Service & control level integration aimed at creating a service standard, which can capture the message exchange patterns, the message types, and the dependencies between the exchanged messages, in terms of both control-flow and data-flow dependencies. The main idea here is to establish an intermediate abstraction layer between the different tools involved in the SDLC and vendors willing to provide integration capabilities in their tools will need to comply to or implement this standard by providing a bridge between the local (technology-specific) application program interface and the (technology-neutral) abstraction layer, in such a way allowing for the definition of work steps beyond the border of the process steps the tool is mainly focused on (Reiss, 1990; 1999; Yap et al., 2005).
  - Ontology & data level integration aimed at defining a shared understanding of the meaning of the heterogeneous data managed by the different tools and in such a way to enable semantic interoperability. Additionally as part of this ontology and data level integration the mapping problem between the original database schemata and the newly defined integrated data model, which becomes more complicated with each with each additional tool, needs to be solved (Emmerich et al., 1997; THOMPSON, 1992).
  - Method & process level integration aimed at defining an "optimal" ALM-process and at ensuring that tools interact effectively towards supporting this process with respect to its steps, constraints and events (I. Thomas & B. Nejmeh, 2002).

- **Enterprise Bus Integration** - An ESB generally provides an abstraction layer on top of an implementation of an enterprise messaging system, which is designed to connect, mediate, and manage interactions between heterogeneous services and applications. To achieve this ESB defines and uses an enterprise message model, which defines the set of messages that the ESB can both transmit and receive in order to encapsulate the functionality offered by its components (the connected tools, applications and services) in a meaningful and in such a way to be able to replace all directed contacts with application,
services and/or tools connected to the bus, so that communication takes place only via the bus. In such a way the problem of integrating different tools can be simplified down to bare routing of messages between the different tools towards an non-prescriptive integration, focusing on sharing the services offered by each tool rather than simply exchanging data among them (Wikipedia, 2011).

**Service-oriented architecture (SOA) based Integration** – SOA enables companies to globally provide reusable services and procedures for agile and flexible business process and tool composition (A. Brown et al., 2005). SOA is based upon three main components: services, which are considered the core of SOA, a service repository, used to make services discoverable and used to provide more information about services and a service bus, used to connect service consumers and service providers (Kohlborn, 2008). In such a way the SOA architecture can be used to facilitate tool integrations in a way similar to the Enterprise Bus Integration, which was described earlier.

**Ontology-based semantic Integration** – Ontologies have first originated in artificial intelligence, but can be applied for tool integration, because they provide a high-level platform independent format for describing data models and for representing the knowledge of a domain in a machine understandable notation, including the relationships between the different concepts. In particular they provide a way to describe the data semantics in a way that allows reasoning and automatic derivation of new knowledge in such a way facilitating the better understanding of the tool underlying data, which is crucial for integration (Hakimpour & Geppert, 2001; Handler, 2011; Rebstock, Engel, & Paulheim, 2008).

**Aspect Oriented Programming (AOP) based Integration** – Using AOP allows a clear logical and semantic separation of cross-cutting concerns from the core functionality of the system and can, in such a way, offer a high degree of flexibility in choosing the implementation technology, when developing reusable workflows. This could be utilized for the purposes of tool integration (Ionescu, Piater, Scheuermann, Laurien, & Iosup, 2009).

- **Getting commitment from the vendors**
  - **EU-funded research project** – Setting up a EU-funded research project to investigate the different already proposed future-suggested design ideas. The a major advantages of using such a project for collaboration is that it allows the partners to share the risks and costs of research between each other and with the EU and offers a neutral terrain, so that the different involved tool vendors need not fear giving up their intellectual property to a third-parity vendor (without getting anything in return), who then uses it to generate profits, as it currently is the cases with the IBM Jazz platform.
  - **Joint Venture** – Creating a separate organization, jointly owned by the research partners is another possibility to improve collaboration by share the risks and costs of research, but it additionally allow more flexibility and can be a good way to avoid interference from the EU.

## 5.5 Identified grand challenges and further research needs

As part of the previously mentioned brainstorming session (see Appendix F) the participants also identified the following grand challenges and further research needs, which depending on the chosen integration strategy might have to be addressed towards constructing the new and improved ALM 2.0 system:

- Develop a better understanding on how to design and create tool integrations, in terms of (Ionescu et al., 2009):
• What kind of data should be integrated and from which sources.
• How the data should be used within the integrated system, in order for it to add value to the chain.
• Better understand and evaluate the possible role of workflow systems in the integration of best-of-breed tools towards ALM 2.0 (Eskeli, 2009).
• Develop best practices and reference architectures for the migration of the still existing monolithic tools, without transparent interfaces and appropriate modularization towards an architecture open for integration (Eskeli, 2009).
• Create a semantic standard (e.g. by the use of ontologies or a common metamodel) to ensure proper function and meaningful data exchange between tools (Marquardt & Nagl, 2003).
• Develop a strategy for successfully solving the syntactic and control issues of tool integrations (Marquardt & Nagl, 2003).
• Develop a strategy on the integration of not only standalone tools, but also of tools, which are deployed as web-services (Neema, 2007).
6 VALIDATION OF RESEARCH RESULTS

The previous chapter presented the research results obtained from focusing on the tasks of identifying the problems with the current ALM tools, eliciting ALM stakeholder requests for future versions of ALM suites and finally making initial design suggestions, which can be used as an initial basis for discussion in a future ALM redesigning process.

This chapter tries to validate the correctness of the achieved research results.

6.1 Validation of requirements elicitation results

IEEE Standard 830-1998 (Society, 1998) defines several characteristics of good requirements. According to this standard a valid or also correct requirement is one that the system shall (must) meet. This means that the requirement has to help to satisfy a stakeholder need by describing something that the system under development must do or a constraint on the way it must be done, rather than trying to achieve some “pie in the sky” objective. In our case, where the users requests are expressed partially as requirements and partially as use cases, this means that the entire specification is correct if, and only if, every single requirement and use case there is one that can be traced back to a stakeholder request coming from the interviews or questionnaires and therefore something the future ALM system shall meet.

According to (Society, 1998) there is no tool or procedure to ensure correctness or validity of a requirements specification. In real development projects it is a common practise to accomplished such a confirmation of correctness by review with the stakeholders who will be directly responsible for the success or failure of the product in the market place.

A requirements review is a working sessions with the basic to ensure the high quality of the requirements and the requirements specification. Requirements review however require a lot of time for preparation and execution and there is a general rule of thumb that during a such a review session only 4 requirements can be reviewed in one hour. In order to increase the work rate the authors facilitated a review session on three levels:

- First the author invited representatives from all stakeholder groups to an online requirements walkthrough session, which gave the author the opportunity to present the specification to all participants by walking them through the document and answering first questions about the document, the entire project and the goals of the following review upfront. The inspection also allowed to uncover initial flaws with the document, like too small and unreadable model screenshots and some formatting issues. These could be solved before the offline review meeting, which took place in the following week and had the goal to validate the correctness of each use case and requirement in the document.

- After the initial walkthrough the author created a separate model view for each different major stakeholder role and extracted a separate sub-document, which only showed the use cases and requirements requested (in the interviews or questionnaires) by the according role. In such a way each participant received a different document with only the use cases and requirements concerning the role he was representing. The participant received three days time for an offline review of their document and to send their comments and questions back to the author for resolution.

- The final live review session took place after all questions and comments from the offline reviews were incorporated in the document and took less than two hours. It was facilitated by the document author and concentrated manly on ensuring the validity and correctness of only those requirements common for all roles and/or where there were still unsolved issues, question, comments with. A few requirements, which could not be traced back to meeting or interview minutes, or to the answers of the questionnaires, were removed entirely from the specification.
6.2 Validation of tool evaluation results

This section discusses the validation of the tool evaluation report, which presents a key part of the chain of quality assurance measures undertaken by the authors to ensure the correctness of the evaluation results. In order to be able to claim some righteousness, honesty and representativeness of the subjectively evaluated (by the authors) ALM tool the results needed to be evaluated by some independent and impartial “body” (none of the identified stakeholders, which all are driven and motivated by their particular interest and agenda), who has some experience in the ALM area. For this purpose the author created a small Focus Group of five people, which met for a 3-hour workshop to review and discuss the evaluation results.

A Focus Group is a popular form of qualitative research, where a group of individuals, in our case with the common characteristic of being tool experts in the area of ALM, is brought together for an open-ended, round-table discussion about an issue. In order to provide some guidance and to make the entire validation procedure more transparent the author created a structured quality assessment framework in the form of a checklist of evaluation criteria. The initial criteria were defined as a result of the identified stakeholder requests and the literature review and analysis of existing tools. To enhance and consolidate the usefulness of the criteria, a small workshop with the experts from the focus group was held. After intensive discussions during the workshop, some criteria were made more concrete, some were split, some were dropped completely, and one new criterion was added. Ten of the criteria were ranked as high-priority and were used to validate the correctness of the tool evaluation results. The following graphic shows the so resulting validation checklist:

<table>
<thead>
<tr>
<th>CriteriaNr</th>
<th>Criteria</th>
<th>Findings</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Interoperability with other tools</td>
<td>P1 P2 P3 P4 P5</td>
</tr>
<tr>
<td>2.</td>
<td>Support for all ALM process areas and perspectives</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>Traceability</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>Reporting</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>Automation</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>Extendibility / Scriptability</td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>Scalability</td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td>Distributed working</td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td>Automatic Data Migration from other tools</td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td>External factors (Price, Loss of Know How, etc.)</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Validation checklist

As a result of the focus group discussion, one tool was entirely dropped, one additional tool was added to the list of evaluated tools, and several findings were revised and restated or rephrased.

The dropped tool was the Eclipse Application Lifecycle Platform (ALF), as the participants noted that the project has been permanently discontinued prior producing any usable deliverables.

Instead of the Eclipse ALF Platform the participants suggested the evaluation of the IBM Jazz platform, which pursues the same goals of creating an open, extensible platform, built the same (eclipse) extensibility model.

Finally, for all evaluated tools the findings, in regards to the specified 10 criteria were made more clear and explicit, in order to make the validation easier for the participants.
6.3 Validation of design ideas

The suggested design ideas for the future ALM tools have all been already applied in different project and for different purposes. Although not all of these different purposes can be directly linked to the posed tool integration dilemma, in the author’s opinion they all present technologically feasible (possible) solutions to the identified heterogeneity problem of tool integration, caused by using multi-databases, different semantics, etc. in the different tools. For solving the previously described problems of today’s ALM tools however it will be necessary to pair the right combination of these suggested integration techniques with the right business model, which invites third party tool vendors to invest and openly collaborate around these suggestions towards achieving the ultimate goal of an open and extensible ALM platform. Finding this combination is by itself a validation of that argument and will be the focus of a future project aimed at creating a full-featured design proposal, validated in an initial tool prototype (see section 7.3 Next Steps).
7 OUTLOOK

This final chapter discusses the results of our work. In this chapter an analysis of our own research is presented in a form of a discussion with regard to theoretical and practical sides of ALM.

7.1 Summary

In past few years Application Lifecycle Management has developed to one of the most sought-after aspects in the software development process. This is due to it’s promise to put everyone involved in the software development process ad beyond (e.g. service management and portfolio management) onto the same (virtual) system (by ensuring a semantic and deep integration of the different tools used across the lifecycle), regardless of their current location or role and ensuring repository neutrality. In such a way ALM will improve a team’s ability to collaborate in context of the work they are doing, to exchange and manage knowledge and information and to respond to changes in requirements, processes, etc. It additionally will provide the necessary end to end traceability and required visibility into the development processes and consequently give answers to the critical questions of who is doing what, how they are doing it, how long it takes, and when goals are being met or missed allowing in such a way to eliminate any wasted duplication of efforts. Further an integrated ALM solution will also allow for automation, to support the imposition of already existing corporate processes, policies and standard as well as best-practices (Magid, 2007).

This work shows ALM in it’s initial form, where the individual vendors have tried to “achieve modicum of Integrated ALM by first integrating their own ALM point tools and then integrating some selection of the popular point tools from other vendors” (Patty Brown, 2011). This ALM architecture, in this work often referred to as point-to-point integration between different practitioner tools, has several heavy limitations, including coping with the complexity of possible tools combinations and therefore tool integrations, which need to be managed and updated. Besides this these kind of integration rarely provides the deep semantic amalgamation between the tools and therefore can not provide the major benefits in regards, to traceability, automation capabilities and reporting capabilities, expected by an ALM solution.

In the past few years ALM has migrated from this initial form to it’s current form of predominantly single vendor Integrated ALM Tools. This form of ALM is characterized by vendors coming up with their own solutions to the point-to-point integration problems by offering their own an integrated ALM. These solutions usually are offered by large tool vendors like IBM, HP, Microsoft, etc., who try to provide the benefits of an ALM system by integrating their various solutions to the different lifecycle phase (e.g. HP LoadRunner, HP Quality Center, etc.). Due to the fact that these different lifecycle phase tools are all made by the same vendor, a deep semantic integration is easier to achieve. While these solutions also aim at including third-party vendor tools, the current implementations on the market do not meet these needs. This means that such ALM solutions have also a few important limitations: on the one side customers face the difficulty of economically justifying the replacement of existing tools by a single vendor tool, a move which normally results in higher costs, rejection by the users, who are used to the existing tool chain and necessity of retraining of the development team members in new tools, which may delay development projects unnecessarily. On the other side highly unlikely that built-in tools from a single vendor can serve the needs of a wide range of development groups, each applying a different development process. As a consequence the users are forced to use these all-in-one
solutions even when better and often less expensive (sometime free open source) tools are available and appropriate for their needs (Patty Brown, 2011).

Due to this limitations of ALM today, this work investigated the actual ALM stakeholder needs. To do so the author first conducted several informal interviews in an industrial partner organization, in order to identify major areas of concern and initial list of relevant stakeholders. Based on this input the author developed a comprehensive questionnaire and interviews aimed to identify the stakeholder’s needs, wishes and requirements for a future ALM system. Based on the received answers the authors performed an analysis, which resulted in a large UML Use Case and Requirements model specifying the relevant aspects a future ALM systems needs to cover. One of the key learning from the analysis of this questionnaire was the insight that ALM systems and ALM integration means need to move from the legacy point-to-point integration and today’s single vendor integrated ALM tools towards a Multi Vendor Best of Breed Integrated ALM Tools, in order to provide simpler development and integration, to protect already done investments in tools, infrastructure and know how and to allow the use of the best tool for the job at hand (Patty Brown, 2011).

The research work, as well as this thesis wrap up with a brainstorming sessions (respectively the documented results of the latter) outlining several viable research topics and/or research topic combinations, which should be further investigated.

7.2 Verification of project goals

Our initial goal was to research the theoretical side of ALM, find the practical hiccups of ALM, and suggest the feasible improvements to the ALM. In order to do that we made a deep theoretical research, diving in the key aspects of ALM, and listing the advantages and disadvantages. We decided to make an interview to seek out what the specialists, working in the field of ALM, have to complain about or suggest. Based on the disadvantages we had to make several propositions regarding what questions we would need to ask the interviewee to uncover the practical leaks of various techniques, tools, and ALM as a whole. Once the interview was done, the brainstorming, and the summarising of the results to make out the suggestions for improvement, took place.

The literature synthesis involved a vast number of different sources, such as books, ebooks, articles, journals, news blogs, etc. Some of the figures were drawn for better explanation of the ALM theoretical background. Thus, the goal of the gathering of the theoretical ALM’s background to make out the disadvantages was achieved.

The goal of finding out the practical difficulties of ALM was achieved by examining several suites’ attributes of tools, and interviewing the specialists in the field of ALM. The intricacies of this goal included the absence of enough material and feedback that could be found on the net. Another difficulty to this goal was the making up of the right questions for the interview. A lot of aspects of ALM had to be taken into account, and several important issues were addressed in the interview. The interview was a bit long, and not all of the participants fully contributed to answering all of the questions. Although it was foreseen, on the other hand it was difficult to avoid the amount of questions, since few, if any, material can be found on the practical sides of ALM, and the questions inevitably needed the answers in order to achieve this goal. The results were more than satisfactory, since we had more participants than we had hoped for, and we received a lot of necessary feedback to make the right conclusions.

The most difficult part of this thesis was the stage once we gathered all the necessary data and texts, and had to decide what can help ALM improve. After the brainstorming, to achieve our goal of suggesting the improvements with relevant arguments, the several proposed solutions were analysed, and some of them, which had few arguments, were left out. The suggested improvements were researched further to some extent in order to make well-reasoned backed-up statements. The main goal of stating the possible solutions was thus achieved.
We believe that we have achieved the initial goals of this thesis, and have made an even deeper unintended research in suggesting what techniques in particular would improve ALM and how, and most importantly the challenges of what these techniques present in themselves were carried out. Additionally we derived what further research should focus on in order to continue to improve ALM in this fashion.

7.3 Next Steps

In the previous chapter, we have shown, that todays ALM tools, which to a large extend are still based on monolithic ALM stacks and designed by a single vendor, or even in the rare cases, where they are based on open for integration ALM platforms, which however are still largely unsupported by the majority of third vendor tools, do not meet or even contradicts the objectives of the end-users, namely quick and flexible as needed integration of best-of-breed tools from the different software lifecycle phases.

In order to achieve this objective, our work focuses on the first phase of every new software development project, namely analyzing the current state-of-the-art (in section tool analysis) and clarifying the requirements and expectations of the different involved stakeholders (in the sections stakeholder analysis and requirements elicitation) and even bridges the gap between requirements and software architecture by suggesting possible design alternatives and ideas (in the section design concepts and ideas), which could solve the current problems and fulfill the stakeholders functional and non-functional requirements.

The logical next step is to evaluate these and other new design concepts and ideas, and combination of those in order to find the optimal design, which best fits the stakeholder needs and will be the major goal of a follow-up project. This project will use this work as an input for writing a research grant proposal for a EU-funded research project involving several large application lifecycle tool vendors in evaluating the different ALM tool integration design options and piloting an initial prototype for such a solution.
REFERENCES


APPENDICES

Appendix A

Basic Definitions

• **Application Lifecycle Management**: ALM “refers to the capability to integrate, coordinate and manage the different phases of the software delivery process. From development to deployment, ALM is a set of pre-defined process and tools that include definition, design, development, testing, deployment and management. Throughout the ALM process, each of these steps are closely monitored and controlled” (Webopedia, 2011a).

• **Software Development Lifecycle (SDLC)**: SDLC is “a structured methodology used in the development of software products and packages. This methodology is used from the conception phase through to the delivery and end of life of a final software product” (Webopedia, 2011b).

• **Service management and operations**: IT Service Management and operations “is a process-based practice intended to align the delivery of information technology (IT) services with needs of the enterprise, emphasizing benefits to customers. ITSM involves a paradigm shift from managing IT as stacks of individual components to focusing on the delivery of end-to-end services using best practice process models” (the leading IT encyclopedia and learning center, 2006). In the specific context of ALM “it focuses on the life of an application or system in a production environment” (Rossberg, 2008).

• **Application Portfolio Management (APM)**: APM “is a framework for managing enterprise IT software applications and software-based services. APM provides managers with an inventory of the company's software applications and metrics to illustrate the business benefits of each application” (the leading IT encyclopedia and learning center, 2011).

• **Tool integration**: Tool Integration refers to “the extent to which tools agree” in relation to “data format, user-interface conventions, use of common functions, or other aspect of tool construction” (Ian Thomas & B. A. Nejmeh, 1992).

• **Traceability**: Traceability refers to “the ability to chronologically interrelate uniquely identifiable entities in a way that is verifiable” and in such a way to ensure the “the completeness of the information about every step in a process chain” (M. Barker, 2007; Wikipedia, 2011m).

• **Automation**: Automation is a “process being automated through the use of computers and computer software. Processes that have been automated require less human intervention and less human time to deliver” (Webopedia, 2011c).

• **Reporting**: in the context of ALM Reporting refers to a number of technical means aimed to “provide visibility into the progress of development efforts” (Rossberg, 2008).
Appendix B
Appendix C
# ALM Survey

This is a survey undertaken with a primary focus on understanding your / your organization's SDLC practices and policies in order to learn your needs and requirements for an ALM solution that can offer you complete integration of your disparate tools (respectively a single tool replacing them) and will also ensure full traceability, real-time visibility into your project and application status, including costs and schedule variances and will support your project management goals!

## General Questions
This section asks some general information questions about you and your organization.

### What role do you play in your organization?
- [ ] Architect (Technical, Infrastructure, Security, Enterprise, etc.)
- [ ] Designer / Developer
- [ ] Business Analyst / Requirements Engineer
- [ ] CIO / IT Director / Board Director
- [ ] Development Manager / Other IT Manager
- [ ] Software Engineer
- [ ] Tester / Test Manager / Test Designer / Test Analyst
- [ ] Configuration Management Engineer
- [ ] Other:  

### Approximately how large is your organization?
- [ ] Less than 10 employees
- [ ] 10 to 49 employees
- [ ] 50 to 249 employees
- [ ] 250 to 4999 employees
- [ ] 5000 to 24999 employees
- [ ] more than 25000 employees

### Where are you based?
- [ ] Europe
- [ ] North America
- [ ] South / Central America
- [ ] Middle East / Africa
- [ ] Central Asia (incl. China and India)
- [ ] Asia-Pacific

## ALM Definitions
This section asks some general questions about your (your organization's) understanding of the ALM
discipline and the ALM tools.

**What is your definition / understanding of ALM, both as a discipline and as a product category?**

*Please give a short definition in form of 2-3 bullet points, explaining the in your opinion main characteristic of an ALM.*


**How relevant / important is ALM for your organization?**

1 2 3 4 5

not relevant [ ] [ ] [ ] [ ] relevant [ ]

**How relevant / important is ALM for your particular role in the organization?**

1 2 3 4 5

not relevant [ ] [ ] [ ] [ ] relevant [ ]

**What particular improvement(s) are you hoping to achieve by applying an ALM solution in your project(s)?**


**Does your organization currently use an ALM solution?**

1 2 3

no [ ] [ ] yes [ ]

**Did your organisation stumble upon any features in ALM that may or do postpone or slow down the product development?**

1 2 3

no [ ] [ ] yes [ ]

**Does your organization currently integrate different tools and components? If yes, which ones?**
How easy is the integration to implement?
1 2 3 4 5
rather difficult ○ ○ ○ ○ very easy

Are you satisfied with the results of the integration on a

<table>
<thead>
<tr>
<th></th>
<th>not satisfied</th>
<th>N/A</th>
<th>satisfied</th>
</tr>
</thead>
<tbody>
<tr>
<td>data and semantic level?</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>process and method level?</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>service and control level?</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

Do you use a simple point-to-point integration between the different tools or some common integration bus and shared information model?
1 2 3
point-to-point ○ ○ ○ integration bus

Please list some of the problems you experienced during the integration setup phase?


Do you satisfied with the availability and performance of such standards and information models or do you believe that this area needs further research?
1 2 3 4 5
not satisfied ○ ○ ○ ○ fully satisfied

**ALM Scope**
This section asks some questions about the, in your opinion, appropriate scope of an ALM solution.

Which statement regarding the appropriate scope for an ALM solution do you agree most with?
- [ ] An ALM solution should have solely an application (in terms of application development lifecycle) focus!
- [ ] It is also (besides the application focus) important to integrate IT operations into an ALM solution,
as this will allow to resolve application (in terms of application development lifecycle) problems and to modify the process around the software delivery according to any changes made in the production environment.

☐ An ALM solution must be flexible and support different levels of business involvement as there is no "one size fits all" solution that is appropriate to all organizations.

☐ Other: 

What do you believe are key components and services, which an ALM solution necessarily needs to integrate for the management and governance of the application lifecycle process?

☐ Ideation / Idea Management Tool
☐ Requirements Elicitation Tool
☐ Requirements Management Tool
☐ Design and Modeling Tool
☐ Integrated Development Environment
☐ Unit Test Tool
☐ Project Management Tool
☐ Document Management Tool
☐ Build Tool
☐ Configuration Management Tool
☐ Test Management Tool
☐ Test Automation Tool
☐ Debugger / Profiler
☐ Issue / Change Management Tool
☐ Helpdesk Tool
☐ Timesheet Tool
☐ Other: 

Do you believe that ALM in all it’s facets is feasible for small and middle-sized companies?

1 2 3 4 5

not feasible ☐ ☐ ☐ ☐ feasible

Do you believe that ALM in all it’s facets is necessary for small and middle-sized companies?

1 2 3 4 5

not necessary ☐ ☐ ☐ ☐ necessary

ALM Integration Level
This section asks some questions about the, in your opinion, appropriate level of integration of an ALM system.
Do you believe that for a tool to be considered a high-quality ALM tool it has to be a complete solution like for instance a suite offered by a single vendor? Or do you believe that it is better for it to be an orchestration and integration of several best-of-breed tools, each focused on the specific functions necessary in a particular lifecycle step?

1 2 3
single-vendor solution ☐ ☐ best-of-breed tool orchestration

If you were given the choice between the introduction of an single vendor ALM solution and a framework allowing deep best-of-breed tool integration, which solution would you prefer?

1 2 3 4 5
single-vendor solution ☐ ☐ ☐ ☐ best-of-breed tool orchestration

Using a complete solution, offered by a single vendor would involve the migration of all data from the different repositories and tools, your organization has already invested in, to the new ALM system. Is your organization prepared to cope with these new costs (financially in terms of new licenses and and in terms of time).

1 2 3 4 5
no ☐ ☐ ☐ ☐ yes

Where across the software delivery lifecycle would you like to see more process support (frameworks, best practices) from an ALM tool?

ALM Features
The following section asks several question about the different SDLC practices in your organization. The information is used to derive necessary features and capabilities of an ALM solution.

Requirements Engineering / Product Management

Do you during the requirements capture activities tend to mix different techniques depending on the nature of your requirements needs, written, use cases, diagrams, etc.?

1 2 3 4 5
no ☐ ☐ ☐ ☐ yes
Do you normally perform some kind of stakeholder analysis and prioritization in your projects and if yes how do you use this information later (e.g. prioritization of requirements, etc.)?

Do you agree that change auditing and notification should be very flexible and carefully controlled, as over-notification is sometimes worse than none at all, resulting in inattention to changes in general?

Do you use impact analysis to track the impact of changes in the requirements on the rest of the system?

Do you plan for reuse of requirements across the different product versions?

Do you explicitly manage variability in your system?

Do you maintain a tight correlation between the work you perform and the initial requirements to ensure that what's delivered is what the user needed and wanted.

To what other artifact do you trace your requirements?
What different types of testing happens?

How is testing measured?

How is testing planned?

What test related roles are there in your organization?

Does your organization use test automation?

Do you consider code coverage?

Are your system tests directly traced to requirements artifacts? Do you use these traces to derive any metrics? (e.g. requirements covered by tests)

Do you propagate risks and priorities identified during the RE activities to the according test artifacts?
Based on what criteria do you plan your regression test suites?

**Agility & Formality**

Are you processes rather flexible and therefore require small and lightweight tools with few restrictions and lots of auditing, or are your processes well tested and proven, and require a tool able to enforce them?

- [ ] flexible
- [ ] rigid

Do all decisions need to be documented?

- [ ] 1
- [ ] 2
- [ ] 3
- [ ] rarely
- [ ] often

Do individuals work on several projects concurrently?

- [ ] 1
- [ ] 2
- [ ] 3
- [ ] 4
- [ ] 5

Are the project's teams geographically dispersed?

- [ ] 1
- [ ] 2
- [ ] 3
- [ ] rarely
- [ ] often

Do the scope and requirements of the solution change a lot?

- [ ] 1
- [ ] 2
- [ ] 3
- [ ] rarely
- [ ] often

Are your projects time driven?

- [ ] 1
- [ ] 2
- [ ] 3
- [ ] rarely
- [ ] often

Do your projects have well defined roles & work streams?

- [ ] 1
- [ ] 2
- [ ] 3
Do you need to accurately track the exact time spend on specific tasks?

1 2 3

no  o  o  yes

Architecture

Does the architecture consider flexibility for change?

1 2 3

no o o o yes

Code Quality

Do you have a well-defined and thorough check-in process which includes quality checks?

1 2 3

no o o o yes

User Interface

Do you use storyboarding?

1 2 3

no o o o yes

Does UI plan consider costs associated with different design options?

1 2 3

no o o o yes

Are user interface designs prototyped before they are implemented?

1 2 3

no o o o yes

Are changed in UI tracked against requirements and builds?

1 2 3

no o o o yes
## Project Management

Is the project's vision and scope well defined and understood and does it map to the business problem?

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td></td>
<td>yes</td>
</tr>
</tbody>
</table>

Do you think that the Customer needs visibility into the project in order to be able to effectively support it?

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td></td>
<td>yes</td>
</tr>
</tbody>
</table>

What metrics do you derive from the project? (e.g. budget, schedule, risk, etc.)

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td></td>
<td>yes</td>
</tr>
</tbody>
</table>

Are risks known and actively managed?

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td></td>
<td>yes</td>
</tr>
</tbody>
</table>

Are the dependencies on third parties and the customer being managed?

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td></td>
<td>yes</td>
</tr>
</tbody>
</table>

Are any specific methodologies and best practices being followed? Do you expect them to be supported out of the box by the ALM solution?

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td></td>
<td>yes</td>
</tr>
</tbody>
</table>

Are resource plans in place and being managed in-line with project scope and budget changes?

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td></td>
<td>yes</td>
</tr>
</tbody>
</table>

Does the development process rely on custom-developed tools?

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td></td>
<td>yes</td>
</tr>
</tbody>
</table>
Configuration Management

Is a build process well defined?

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>

Is there a regular build schedule?

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>

Is an automated build verification process in place?

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>

Is there an effective build failure/success notification process in place?

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>

Other

Do you believe that there are specific aspects of automation that should be sought out when looking at today's ALM tools? Which ones?

People

The following section asks several question about the different people involved in the SDLC activities.

What roles need to be supported by the new ALM solution?

Do you believe that using a single tool for all the different roles in the ALM can lead to low productivity, due to increased tool complexity.
Do you believe that role misalignment can lead to overspending on ALM licensing?

1 2 3

no □ □ □ yes
General Questions
This section asks some general information questions about you and your organization.

**What role do you play in your organization?**
- **Architect (Technical, Infrastructure, Security, Enterprise, etc.)**: 2 (13%)
- **Designer / Developer**: 3 (19%)
- **Business Analyst / Requirements Engineer**: 2 (13%)
- **CIO / IT Director / Board Director**: 0 (0%)
- **Development Manager / Other IT Manager**: 4 (25%)
- **Software Engineer**: 3 (19%)
- **Tester / Test Manager / Test Designer / Test Analyst**: 7 (44%)
- **Configuration Management Engineer**: 1 (6%)
- **Other**: 0 (0%)

People may select more than one checkbox, so percentages may add up to more than 100%.

**Approximately how large is your organization?**
- **Less than 10 employees**: 2 (11%)
- **10 to 49 employees**: 1 (6%)
- **50 to 249 employees**: 3 (17%)
- **250 to 4999 employees**: 4 (22%)
- **5000 to 24999 employees**: 6 (33%)
- **more than 25000 employees**: 0 (0%)

**Where are you based?**
- **Europe**: 6 (33%)
- **North America**: 7 (39%)
- **South / Central America**: 3 (17%)
- **Middle East / Africa**: 0 (0%)
- **Central Asia (incl. China and India)**: 0 (0%)
- **Asia-Pacific**: 0 (0%)

ALM Definitions
This section asks some general questions about your (your organization's) understanding of the ALM discipline and the ALM tools.

**What is your definition / understanding of ALM, both as a discipline and as a product category?**
ALM an orchestration of tools and of processes process, which have the goal to allow a seaming-less integration and governing of the different phases in the software development lifecycle like planning, definition, design, development, testing, deployment, and management. ALM is the coordination (based on different tools and
How relevant / important is ALM for your organization?

1 - not relevant 0 0%
2 - 0 0%
3 - 0 0%
4 - 2 11%
5 - relevant 14 78%

How relevant / important is ALM for your particular role in the organization?

1 - not relevant 0 0%
2 - 0 0%
3 - 0 0%
4 - 0 0%
5 - relevant 16 89%

What particular improvement(s) are you hoping to achieve by applying an ALM solution in your project(s)?

Ratify Best practice Processes Gain mode insight into the entire process Improve collaboration Increase quality and productivity Manage change Allow "plug and play" integration of tool without data or semantic loss.

Does your organization currently use an ALM solution?

1 - no 0 0%
2 - 5 28%
3 - yes 11 61%

Did your organisation stumble upon any features in ALM that may or do postpone or slow down the product development?

1 - no 0 0%
2 - 3 17%
3 - yes 13 72%

Does your organization currently integrate different tools and components? If yes, which ones?

different IDE's with Collaboration tools, Bug tracking tools and Excel (where we keep requirements) different IDE's with Collaboration tools, Bug tracking tools and Excel (where we keep requirements) yes, IBM DOORS with Borland Together, TestBench, IBM ClearCase, Visual Studio yes, IBM DOORS with Borland Together, TestBench, IBM ClearCase, Visual Studio yes, modeling tools with RMDBs and issue tracking tools yes, modelling tools with RMDBs and issue tracking tools and configuration management tools RMDB & Test Management Tools IDEs & Configuration Management Systems & Continuous Integration ...

How easy is the integration to implement?

1 - rather difficult 10 56%
2 - 5 28%
3 - 0 0%
4 - 1 6%
5 - very easy 0 0%
Are you satisfied with the results of the integration on a - data and semantic level?

- not satisfied: 13 (72%)
- N/A: 0 (0%)
- satisfied: 1 (6%)

Are you satisfied with the results of the integration on a - process and method level?

- not satisfied: 5 (28%)
- N/A: 6 (33%)
- satisfied: 3 (17%)

Are you satisfied with the results of the integration on a - service and control level?

- not satisfied: 6 (33%)
- N/A: 7 (39%)
- satisfied: 1 (6%)

Do you use a simple point-to-point integration between the different tools or some common integration bus and shared information model?

- point-to-point: 1 (67%)
- integration bus: 2 (17%)
- 3: 1 (6%)

Please list some of the problems you experienced during the integration setup phase?

- data loss round-trip engineering impossible
- data loss round-trip engineering impossible
- semantic loss very limited automation support
- data redundancy manually triggered and no life synchronization available
- semantic loss very limited automation support
- data redundancy manually triggered and no life synchronization available
- Integration is manual and very error prone.
- There is a lot of data and semantic loss when transferring information from one tool to another. Data is redundant in the different tools and difficult to keep up-to-date.
- Automation beyond the border of the single tools is v

Do you satisfied with the availability and performance of such standards and information models or do you believe that this area needs further research?

- not satisfied: 5 (28%)
- 2: 8 (44%)
- 3: 3 (17%)
- 4: 0 (0%)
- fully satisfied: 0 (0%)
**ALM Scope**

This section asks some questions about the, in your opinion, appropriate scope of an ALM solution.

Which statement regarding the appropriate scope for an ALM solution do you agree most with?

- An ALM solution should have solely an application (in terms of application development lifecycle) focus!
- It is also (besides the application focus) important to integrate IT operations into an ALM solution, as this will allow companies to fine-tune processes and to modify the process around the software delivery according to any changes made in the production environment.
- An ALM solution must be flexible and support different levels of business involvement as there is no “one size fits all” solution that is appropriate to all organizations.

People may select more than one checkbox, so percentages may add up to more than 100%.

What do you believe are key components and services, which an ALM solution necessarily needs to integrate for the management and governance of the application lifecycle process?

- Ideation / Idea Management Tool
- Requirements Elicitation Tool
- Requirements Management Tool
- Design and Modeling Tool
- Integrated Development Environment
- Unit Test Tool
- Project Management Tool
- Document Management Tool
- Build Tool
- Configuration Management Tool
- Test Management Tool
- Test Automation Tool
- Debugger / Profiler
- Issue / Change Management Tool
- Helpdesk Tool
- Timesheet Tool
- Other

People may select more than one checkbox, so percentages may add up to more than 100%.

Do you believe that ALM in all it’s facets is feasible for small and middle-sized companies?

- 1 - not feasible
- 2
- 3
- 4
- 5 - feasible

People may select more than one checkbox, so percentages may add up to more than 100%.

---

**Not satisfied**

**Fully satisfied**

---
Do you believe that ALM in all its facets is necessary for small and middle-sized companies?

1 - not necessary 0 0%
2 2 11%
3 0 0%
4 0 0%
5 - necessary 0 0%

ALM Integration Level

This section asks some questions about the, in your opinion, appropriate level of integration of an ALM system.

Do you believe that a tool to be considered a high-quality ALM tool it has to be a complete solution like for instance a suite offered by a single vendor?
Or do you believe that it is better for it to be an orchestration and integration of several best-of-breed tools, each focused on the specific functions necessary in a particular lifecycle step?

1 - single-vendor solution 0 0%
2 8 44%
3 - best-of-breed tool orchestration 8 44%

If you were given the choice between the introduction of a single vendor ALM solution and a framework allowing deep best-of-breed tool integration, which solution would you prefer?

1 - single-vendor solution 0 0%
2 0 0%
3 3 17%
4 5 28%
5 - best-of-breed tool orchestration 8 44%

Using a complete solution, offered by a single vendor would involve the migration of all data from the different repositories and tools, your organization has already invested in, to the new ALM system. Is your organization prepared to cope with these new costs (financially in terms of new licenses and and in terms of time).

1 - no 7 39%
2 1 6%
3 5 28%
4 3 17%
5 - yes 0 0%

Where across the software delivery lifecycle would you like to see more process support (frameworks, best practices) from an ALM tool?

Mostly in the interfaces between the different fields, e.g. RM & Architecture, RM & Test, RM & Variability Management, Variability Management & Architecture, Variability Management & Test, Development & Architecture, Development & Test. The interfaces between the different fields are currently the soft spot in the ALM process, because the different tools are not designed to easy interact with each other.

generation of test cases out of models project reviews, generation of test cases out of ...

ALM Features

The following section asks several question about the different SDLC practices in your organization. The information is used to derive necessary features and capabilities of an ALM solution.
Requirements Engineering / Product Management

Do you during the requirements capture activities tend to mix different techniques depending on the nature of your requirements needs, written, use cases, diagrams, etc.?  
1. no 0 0%  
2. 0 0%  
3. 0 0%  
4. 7 39%  
5. yes 9 50%  

Do you normally perform some kind of stakeholder analysis and prioritization in your projects and if yes how do you use this information later (e.g. prioritization of requirements, etc.)?  
yes, stakeholders are late traced to requirements and use cases and stakeholder priorities are used to prioritize requirements  
yes, stakeholders are later associated with use cases and requirements priorities are partially derived from the stakeholder priorities.  
yes, stakeholders are later associated with use cases and requirements priorities are partially derived from the stakeholder priorities.  
Yes, the stakeholders are assigned to use cases and requirements. No prior ...  

Do you agree that change auditing and notification should be very flexible and carefully controlled, as over-notification is sometimes worse than none at all, resulting in inattention to changes in general?  
1. not at all 0 0%  
2. 0 0%  
3. 0 0%  
4. 4 22%  
5. yes, absolutely 12 67%  

Do you use impact analysis to track the impact of changes in the requirements on the rest of the system?  
1. no 0 0%  
2. 0 0%  
3. yes 16 89%  

Do you plan for reuse of requirements across the different product versions?  
1. no 3 17%  
2. 0 0%  
3. yes 13 72%  

Do you explicitly manage variability in your system?  
1. no 3 17%  
2. 0 0%  
3. yes 13 72%
Do you maintain a tight correlation between the work you perform and the initial requirements to ensure that what’s delivered is what the user needed and wanted.

<table>
<thead>
<tr>
<th></th>
<th>no</th>
<th>yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>16</td>
</tr>
</tbody>
</table>

To what other artifact do you trace your requirements?

- tests, issues, architectural models
- test cases, architectural models, risks, stakeholders, issues, tasks
- test cases, architectural models, risks, stakeholders, issues, tasks, product configurations, tests, models, issues, risks, design artefacts
- test cases, architectural models, risks, stakeholders, issues, tasks, product configurations, tests, models, issues, risks, design artefacts, value/effort matrix

Testing

What different types of testing happens?

- unit testing integration testing system testing
- unit testing integration testing system testing acceptance testing
- unit testing integration testing system testing acceptance testing

How is testing measured?

- effort and time
- customer satisfaction index
- delivered defect quantities
- responsiveness (turnaround time)
- product volatility
- defect ratios
- defect removal efficiency
- complexity of delivered product test coverage
- cost of defects
- costs of quality activities
- re-work reliability
- schedule
- Functionality
- code coverage
- problems
- test cases

How is testing planned?

- unit testing is planned as part of the development activities Integration & System testing is planned by the test manger, based on available time and associated by the test case priorities (derived from risks, etc.)
- unit testing is planned as part of the development activities Integration & System testing is planned by the test manger, based on available time and associated by the test case priorities (derived from risks, etc.) by an system analyst considering on the project constraints by an system analyst considering on the project constraints A test manager defines the following parameters bas ...

What test related roles are there in your organization?

- Tester, Test Manager
- Tester User Acceptance
- Tester Test Developer
- Test Analyst
- Test Manager
- Test Manger, Test Analysts, Test Developer, Tester
- Test Manger, Test Analysts, Test Developer, Tester, Performance Tester, User Interface Tester
- Test Manger, Tester, Test Analyst, Test Manager
- Test Manger, Test Analyst, Test Developer, Tester, Test Manager, Test Analyst, Tester
- Test Manger, Test Analyst, Test Developer, Tester, Test Manager, Test Analyst, Tester
- Test Manger, Test Analyst, Test Developer, Tester, Test Manager, Test Analyst, Tester
- test manager, test developer, test analyst, test designer, tester ...

Does your organization use test automation?

<table>
<thead>
<tr>
<th></th>
<th>no</th>
<th>yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>16</td>
</tr>
</tbody>
</table>

Do you consider code coverage?

<table>
<thead>
<tr>
<th></th>
<th>no</th>
<th>yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>16</td>
</tr>
</tbody>
</table>
Are your system tests directly traced to requirements artifacts? Do you use these traces to derive any metrics? (e.g. requirements covered by tests)

- **No**: 0 (0%)
- **Yes**: 16 (89%)

Do you propagate risks and priorities identified during the RE activities to the according test artifacts?

- **No**: 0 (0%)
- **Yes**: 15 (83%)

Based on what criteria do you plan your regression test suites?

- **Time & Value**: 0 (0%)
- **Time Severity Impact & Business Risk**: 11 (67%)
- **Risk, Impact, Cost, Time**: 2 (13%)
- **Technical Evaluation of the Change (Functional and Physical)**: 0 (0%)
- **Criticality of Functions to be Re-tested**: 0 (0%)
- **Impact, Associated Costs and Business Risks**: 0 (0%)
- **Business Risks & Time (Cost)**: 0 (0%)
- **Business Risk & Time (Cost)**: 0 (0%)
- **Business Risk & ...**: 0 (0%)

Agility & Formality

Are you processes rather flexible and therefore require small and lightweight tools with few restrictions and lots of auditing, or are your processes well tested and proven, and require a tool able to enforce them?

- **Flexible**: 14 (78%)
- **Tested and Proven**: 1 (6%)
- **Both**: 3 (17%)

Do all decisions need to be documented?

- **No**: 8 (44%)
- **Yes**: 8 (44%)
<table>
<thead>
<tr>
<th>Question</th>
<th>1 - rarely</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5 - often</th>
<th>Total</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do individuals work on several projects concurrently?</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>18</td>
<td>44%</td>
</tr>
<tr>
<td>Are the project's teams geographically dispersed?</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td></td>
<td></td>
<td>16</td>
<td>89%</td>
</tr>
<tr>
<td>Do the scope and requirements of the solution change a lot?</td>
<td>4</td>
<td>2</td>
<td>10</td>
<td></td>
<td></td>
<td>16</td>
<td>56%</td>
</tr>
<tr>
<td>Are your projects time driven?</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td></td>
<td></td>
<td>16</td>
<td>89%</td>
</tr>
<tr>
<td>Do your projects have well defined roles &amp; work streams?</td>
<td>0</td>
<td>2</td>
<td>14</td>
<td></td>
<td></td>
<td>16</td>
<td>78%</td>
</tr>
<tr>
<td>Do you need to accurately track the exact time spend on specific tasks?</td>
<td>2</td>
<td>0</td>
<td>14</td>
<td></td>
<td></td>
<td>16</td>
<td>78%</td>
</tr>
</tbody>
</table>
Architecture

Does the architecture consider flexibility for change?

- no: 0 (0%)
- yes: 16 (89%)

Code Quality

Do you have a well-defined and thorough check-in process which includes quality checks?

- no: 0 (0%)
- yes: 15 (83%)

User Interface

Do you use storyboarding?

- no: 4 (22%)
- yes: 12 (67%)

Does UI plan consider costs associated with different design options?

- no: 9 (50%)
- yes: 5 (28%)

Are user interface designs prototyped before they are implemented?

- no: 4 (22%)
- yes: 12 (67%)
Are changed in UI tracked against requirements and builds?

<table>
<thead>
<tr>
<th></th>
<th>no</th>
<th>yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>33%</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>56%</td>
</tr>
</tbody>
</table>

Project Management

Is the project’s vision and scope well defined and understood and does it map to the business problem?

<table>
<thead>
<tr>
<th></th>
<th>no</th>
<th>yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>11%</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>78%</td>
</tr>
</tbody>
</table>

Do you think that the Customer needs visibility into the project in order to be able to effectively support it?

<table>
<thead>
<tr>
<th></th>
<th>no</th>
<th>yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>3</td>
<td>16</td>
<td>89%</td>
</tr>
</tbody>
</table>

What metrics do you derive from the project? (e.g. budget, schedule, risk, etc.)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>schedule</td>
<td>risk</td>
<td>budget</td>
</tr>
<tr>
<td>schedule</td>
<td>risk</td>
<td>budget</td>
</tr>
<tr>
<td>schedule, effort, cost, productivity rates, error rates, risks</td>
<td>schedule, effort, cost, productivity rates, error rates, risks</td>
<td></td>
</tr>
<tr>
<td>tasks, schedules, performance, time to complete, costs, etc.</td>
<td>tasks, schedules, performance, time to complete, costs, etc.</td>
<td></td>
</tr>
</tbody>
</table>

Are risks known and actively managed?

<table>
<thead>
<tr>
<th></th>
<th>no</th>
<th>yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>11%</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>78%</td>
</tr>
</tbody>
</table>
Are the dependencies on third parties and the customer being managed?

1. No 3 (17%)
2. 0 (0%)
3. Yes 13 (72%)

Are any specific methodologies and best practices being followed? Do you expect them to be supported out of the box by the ALM solution?

1. No 0 (0%)
2. 0 (0%)
3. Yes 16 (89%)

Are resource plans in place and being managed in-line with project scope and budget changes?

1. No 2 (11%)
2. 2 (11%)
3. Yes 12 (67%)

Does the development process rely on custom-developed tools?

1. No 1 (6%)
2. 0 (0%)
3. Yes 15 (83%)

Configuration Management

Is a build process well defined?

1. No 0 (0%)
2. 0 (0%)
3. Yes 16 (89%)

Is there a regular build schedule?

1. No 0 (0%)
2. 0 (0%)
3. Yes 16 (89%)
Is an automated build verification process in place?

- No: 0, 0%
- Yes: 16, 89%

Is there an effective build failure/success notification process in place?

- No: 0, 0%
- Yes: 14, 78%

Other

Do you believe that there are specific aspects of automation that should be sought out when looking at today’s ALM tools? Which ones?

- Automatic test derivation out of existing models
- Better integration between RE & Test
- Better integration between RE & Variability Management
- Better integration between RE & Product Management
- Better integration between Development & Test
- Better integration between Development & Architecture
- Better integration between Development & Architecture
- Traceability

People

The following section asks several questions about the different people involved in the SDLC activities.

What roles need to be supported by the new ALM solution?

- Architect (Technical, Infrastructure, Security, Enterprise, etc.)
- Designer / Developer Business Analyst / Requirements Engineer CIO / IT Director / Board Director Development Manager / Other IT Manager Software Engineer Tester / Test Manager / Test Designer / Test Analyst Configuration Management Engineer
- Architect (Technical, Infrastructure, Security, Enterprise, etc.)
- Designer / Developer Business Analyst / Requirements Engineer CIO / IT Director / Board Director Development Manager / Other IT Manager Software Engineer Tester / Test Manager / Test Designer / Test Analyst Configuration Management Engineer

Do you believe that using a single tool for all the different roles in the ALM can lead to low productivity, due to increased tool complexity?

- Rather unlikely: 0, 0%
- Likely: 3, 17%
- Very likely: 13, 72%
Do you believe that role misalignment can lead to overspending on ALM licensing?

1. no 0 0%
2. 2 11%
3. yes 14 78%

Number of daily responses

<table>
<thead>
<tr>
<th>Date</th>
<th>Number of Responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>4/8/2011</td>
<td>4</td>
</tr>
<tr>
<td>4/10/2011</td>
<td>3</td>
</tr>
<tr>
<td>4/12/2011</td>
<td>2</td>
</tr>
<tr>
<td>4/14/2011</td>
<td>1</td>
</tr>
</tbody>
</table>
Appendix E
Use Case Document

<table>
<thead>
<tr>
<th>Version:</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created:</td>
<td>12/04/2011</td>
</tr>
<tr>
<td>Last Update:</td>
<td>29/04/2011</td>
</tr>
<tr>
<td>Print Date:</td>
<td>5/05/2011</td>
</tr>
<tr>
<td>By:</td>
<td>Georgi A. Markov</td>
</tr>
<tr>
<td>Distribution:</td>
<td></td>
</tr>
</tbody>
</table>
# Table of Contents

Use Case Document........................................................................................................1

Use Cases .........................................................................................................................8

1 Model.............................................................................................................................8

1.1 Requirements Model .................................................................................................8

1.1.1 <anonymous> ........................................................................................................8

1.1.2 Constraints ............................................................................................................9

1.1.3 Functional Requirements .......................................................................................9

1.1.3.1 Note ....................................................................................................................9

1.1.4 Non-Functional Requirements ..............................................................................9

1.1.4.1 Note ...................................................................................................................10

1.1.4.2 Accessibility .......................................................................................................10

1.1.4.2.1 Remote Access ...............................................................................................11

1.1.4.3 Availability .......................................................................................................11

1.1.4.3.1 High Availability ............................................................................................12

1.1.4.4 Compatibility ......................................................................................................12

1.1.4.4.1 ALM Integration bus .......................................................................................13

1.1.4.4.2 Common information model ..........................................................................13

1.1.4.5 Extensibility .......................................................................................................13

1.1.4.5.1 Modifiability ....................................................................................................14

1.1.4.5.2 Tool integration ...............................................................................................14

1.1.4.5.3 Underlying development model .....................................................................15

1.1.4.6 Maintainability ..................................................................................................15

1.1.4.6.1 Adapt to changes ............................................................................................15

1.1.4.7 Performance .....................................................................................................16

1.1.4.7.1 Performance under load ................................................................................16

1.1.4.7.2 Simultaneous user sessions ..........................................................................16

1.1.4.8 Scalability .........................................................................................................16

1.1.4.8.1 Integration scalability ....................................................................................17

1.1.4.9 Security .............................................................................................................17

1.1.4.9.1 Access rights ...................................................................................................18

1.1.4.9.2 Roles support ................................................................................................18

1.1.4.10 Usability ..........................................................................................................18

1.1.4.10.1 Easy to learn .................................................................................................19

1.1.4.10.2 Easy to use ..................................................................................................19

1.2 Use Case Model .........................................................................................................19

1.2.1 <anonymous> .......................................................................................................20

1.2.2 Actors ....................................................................................................................20

1.2.2.1 Application architect .........................................................................................21

1.2.2.2 Application life-cycle related roles ....................................................................21

1.2.2.3 Architecture phase related roles .......................................................................21

1.2.2.4 Board director ...................................................................................................21

1.2.2.5 Business Analyst ...............................................................................................21

1.2.2.6 C* level manager (CIO, CTO, etc.) ..................................................................22
Primary Use Cases

Common Use Cases

1.2.3.1 Automate tasks ................................................................. 28
1.2.3.2 Collaborate with other users ............................................. 28
1.2.3.3 Define user roles and permissions .................................... 28
1.2.3.4 Filter & Search for information ....................................... 29
1.2.3.5 Perform a Check-in/Check-out ....................................... 29
1.2.3.6 Review other SDLC artifacts (requirements, test cases, change requests, documents) .................................................... 29
1.2.3.7 Share calendars, information, alerts, status, contacts ........ 29
1.2.4 Primary Use Cases .............................................................. 29
1.2.4.1 Application portfolio management relevant UCs

1.2.4.1.1 Create and document a road map that outlines and prioritizes overhaul

1.2.4.1.2 Define & document enterprise-wide standards, policies and reference models

1.2.4.1.3 Develop and document a portfolio management and IT strategy

1.2.4.1.4 Manage the company application portfolio

1.2.4.2 Customer related UCs

1.2.4.2.1 Access customer view on the projects

1.2.4.2.2 Communicate and collaborate with project managers, etc.

1.2.4.2.3 Create change request(s)

1.2.4.2.4 Create dashboards and real-time update's relevant to customer

1.2.4.2.5 Create project reports

1.2.4.2.6 Give feedback and escalate problems

1.2.4.2.7 Participate in review and prototyping sessions

1.2.4.2.8 Receive alerts, review invitations, etc.

1.2.4.2.9 Report problems and seek help

1.2.4.2.10 Review real-time project status

1.2.4.3 Management related UCs

1.2.4.3.1 Assign requirements to iteration

1.2.4.3.2 Assign requirements to resources

1.2.4.3.3 Assign tasks to people

1.2.4.3.4 Create project plans and decompose requirements into project tasks and deliverables

1.2.4.3.5 Create project reports

1.2.4.3.6 Document important decisions

1.2.4.3.7 Manage product variabilities and product lines

1.2.4.3.8 Manage project and business goals

1.2.4.3.9 Manage project risks

1.2.4.3.10 Organize and manage resources

1.2.4.3.11 Set and measure key performance indicator

1.2.4.4 Production environment role related UCs

1.2.4.4.1 Automate IT operation(s)

1.2.4.4.2 Coordinate communication

1.2.4.4.3 Create and document a service strategy

1.2.4.4.4 Create and use a Service Desk

1.2.4.4.5 Create product trainings and make it available to the product users

1.2.4.4.6 Define and assess measures for IT Services

1.2.4.4.7 Design and test IT Services

1.2.4.4.8 Document and manage Product Knowledge

1.2.4.4.9 Performing first-line product diagnosis

1.2.4.4.10 View call and problem logs

1.2.4.5 SDCL role related UCs
1.2.4.5.1 Architecture role related UCs ........................................38
  1.2.4.5.1.1 Assign tasks to developers ........................................39
  1.2.4.5.1.2 Create & maintain traceability ..................................39
  1.2.4.5.1.3 Create architectural models ....................................39
  1.2.4.5.1.4 Create architectural specifications ................................39
  1.2.4.5.1.5 Document architecture decisions ..................................40
  1.2.4.5.1.6 Oversee and manage the development process ..................40
  1.2.4.5.1.7 Understand market trends .........................................40
  1.2.4.5.1.8 Understand the stakeholder's needs and constraints ............40

1.2.4.5.2 Configuration management role related UCs ..........................40
  1.2.4.5.2.1 Build system ..........................................................41
  1.2.4.5.2.2 Create & document a CM plan and assign responsibilities ....41
  1.2.4.5.2.3 Create and document procedures and standards for evolving software ..........................................................41
  1.2.4.5.2.4 Create, manage branch or baseline ................................42
  1.2.4.5.2.5 Display derivation history .........................................42
  1.2.4.5.2.6 Enforce the CM procedures and standards (naming schemes, etc.) ..........................................................42
  1.2.4.5.2.7 Identify the current combinations of components ................42
  1.2.4.5.2.8 Plan and distribute new system releases .........................42

1.2.4.5.3 Development role related UCs .........................................42
  1.2.4.5.3.1 Allocate Requirements to Design Artifacts ......................43
  1.2.4.5.3.2 Allocate time to task .................................................43
  1.2.4.5.3.3 Automatically derive / update project metrics based on the progress report .................................................43
  1.2.4.5.3.4 Create progress report on assigned tasks .......................43
  1.2.4.5.3.5 Create system design ................................................44
  1.2.4.5.3.6 Implement required functionality ..................................44

1.2.4.5.4 Process related UCs ....................................................44
  1.2.4.5.4.1 Customize the system according to the chosen development methodology ..................................................44
  1.2.4.5.4.2 Customize the system according to the chosen development methodology ..................................................44

1.2.4.5.5 RE role related UCs .....................................................45
  1.2.4.5.5.1 Create baselines ......................................................45
  1.2.4.5.5.2 Create goal model ....................................................45
  1.2.4.5.5.3 Create update traces ................................................46
  1.2.4.5.5.4 Create variability model ............................................46
  1.2.4.5.5.5 Elicit, define and manage requirements-relevant information 46
  1.2.4.5.5.6 Generate Specifications .............................................46
  1.2.4.5.5.7 Perform impact analysis after change requests ................46
  1.2.4.5.5.8 Propagate (changed, new) status and priorities to other artifacts 46

1.2.4.5.6 Test related UCs .......................................................46
  1.2.4.5.6.1 Assign resources (tester, time frame, etc.) to test case ...47
1.2.4.5.6.2 Automatically create bug reports and tasks for developers based on test failures .................................................................47
1.2.4.5.6.3 Automatically run test script runs and reports results ......47
1.2.4.5.6.4 Create & manage tests and link them to their corresponding work items, requirements, etc. .........................................................47
1.2.4.5.6.5 Create regression test suite based on different criteria and/or criteria combinations .................................................................48
1.2.4.5.6.6 Create, Plan & run tests ..............................................................................48
1.2.4.5.6.7 Generate test case(s) from UML model........................................48
1.2.4.5.6.8 Outline and document quality assurance procedures......48
1.2.4.5.6.9 Report and/or track testing progress and status of bugs ...48
1.2.4.5.6.10 Submit bugs ..........................................................................................48
1.2.4.5.7 UI design role related UCs .................................................................48
1.2.4.5.7.1 Create a cost-effort estimate for a design option ..............49
1.2.4.5.7.2 Create a storyboard ............................................................................49
1.2.4.5.7.3 Create a workflow model .................................................................49
1.2.4.5.7.4 Create and document UI Design ..................................................49
1.2.4.5.7.5 Trace UI components and changes to UI components to other artefacts ........................................................................50
1.2.4.5.8 Variability Management .................................................................50
1.2.4.5.8.1 Define and document a strategic for systematic reuse ......50
1.2.4.5.8.2 Define and manage application variability ......................................51
1.2.4.5.8.3 Identify and manage common and variable assets ..........51
Use Cases

1 Model

1.1 Requirements Model

The Non-Functional Requirements package specifies the various operational parameters that define the environment in which the system will exist. These are criteria which define performance levels, scalability, security requirements, backup, disaster recovery and other operational requirements.
1.1.2 Constraints

![Diagram of Constraints]

Name: Constraints  
Package: Constraints  
Version: 1.0  
Author: Georgi Markov

Figure 2: Constraints

1.1.3 Functional Requirements

![Diagram of Functional Requirements]

Name: Functional Requirements  
Package: Functional Requirements  
Version: 1.0  
Author: Georgi Markov

Figure 3: Functional Requirements

1.1.3.1 Note

Functional Requirements describe the features, behavior, business rules and general functionality that the proposed system must support.

1.1.4 Non-Functional Requirements
1.1.4.1  **Note**  
These packages contain non-functional requirements specified for the new system. These typically describe performance criteria, reliability, security and other operational parameters.

1.1.4.2  **Accessibility**
1.1.4.2.1 Remote Access

The system shall provide a web interface or another dedicated client program by which it can be easily accessed by remote users.

1.1.4.3 Availability
1.1.4.3.1 **High Availability**

Relevant system services like shall have a minimum uptime of 99% excluding time pre-scheduled for maintenance and/or upgrades.

1.1.4.4 **Compatibility**
The system shall be compatible with every tool implementing the (to be defined and vendor independent) ALM integration bus.

1.1.4.4.1 **ALM Integration bus**

The system shall be compatible with every tool implementing the (to be defined and vendor independent) ALM integration bus.

1.1.4.4.2 **Common information model**

The system shall be able to integrate with every tool, implementing the (to be defined and vendor independent) common information model.

1.1.4.5 **Extensibility**
1.1.4.5.1  **Modifiability**

The system shall be highly customizable and modifiable in terms of both offered functionality and look and feel.

1.1.4.5.2  **Tool integration**

The system shall be able to easily integrate any tool implementing the to be defined ALM integration bus and the common information model, in such a way
allowing to increase/descrease the functionality of the core system

1.1.4.5.3 Underlying development model
The system shall be easily adjustable to the different development process models.

1.1.4.6 Maintainability

Figure 9: Maintainability

1.1.4.6.1 Adapt to changes
The system shall be adaptable to, and allow for changes in the common information model and integration bus.
1.1.4.7 Performance

custom Performance

| Name:       | Performance |
| Package:    | Performance |
| Version:    | 1.0         |
| Author:     | Georgi Markov |

Simultaneous user sessions

notes
The system shall allow up to 1000 simultaneous sessions at any given time.

Performance under load

notes
The system shall an acceptable response time of at the most 5 seconds each - for data search and retrieval, report generation, and other transaction-heavy activities.

Figure 10: Performance

1.1.4.7.1 Performance under load

The system shall an acceptable response time of at the most 5 seconds each - for data search and retrieval, report generation, and other transaction-heavy activities.

1.1.4.7.2 Simultaneous user sessions

The system shall allow up to 1000 simultaneous sessions at any given time.

1.1.4.8 Scalability
1.1.4.8.1 **Integration scalability**

The system shall be scalable to accommodate an unrestricted number of "best-of-breed" tools, as long as those implement the common information model and the integration bus.

1.1.4.9 **Security**
1.1.4.9.1 **Access rights**

The system shall support different access rights (e.g. view, read, write, etc.) in a object level.

1.1.4.9.2 **Roles support**

The system shall support different roles, each associated with different access right to them.

1.1.4.10 **Usability**
1.1.4.10.1 Easy to learn
The system shall be easy to learn.

1.1.4.10.2 Easy to use
The system shall be intuitive and easy to learn.

1.2 Use Case Model
1.2.1 <anonymous>

Actors are the users of the system being modeled. Each Actor will have a well-defined role, and in the context of that role have useful interactions with the system.

A person may perform the role of more than one Actor, although they will only assume one role during one use case interaction.

An Actor role may be performed by a non-human system, such as another computer program.

1.2.2 Actors

Figure 14: Use Case Model
1.2.2.1 **Application architect**

The application architect is responsible for establishing, implementing, communicating, and maintaining the application portfolio within the IT environment, for developing and maintaining an application roadmap plans aimed at supporting the needs across the company and finally for establishing architecture standards for application documentation in the organization.

1.2.2.2 **Application life-cycle related roles**

1.2.2.3 **Architecture phase related roles**

1.2.2.4 **Board director**

The Board director is responsible for ongoing governance of the entire organization.

1.2.2.5 **Business Analyst**

The business analyst is responsible for identifying the business needs of the organizations clients and stakeholders in order to help determining the solutions to business problems. He is also responsible for requirements elicitation, analyses, validation, documentation and requirements management of business, organizational and/or operational requirements.
1.2.2.6  **C* level manager (CIO, CTO, etc.)**
The C*-level manager is responsible for overall leadership, planning, development, and management of (in the case of CTO, information technology) resources across an organization.

1.2.2.7  **Change manager**
The Change Manager is responsible for the success of the change management strategy and implementation for the project.

1.2.2.8  **Configuration management engineer**
The Configuration Management Engineer is responsible for software configuration and release management activities for all software.

1.2.2.9  **Configuration management related roles**

1.2.2.10  **Customer**
Customer refers to a current or potential buyer or user of the products or services provided by the Organization.

1.2.2.11  **Data Architect**

1.2.2.12  **Database administrator**
The DBA is responsible for the design, implementation, maintenance and repair of the organization's or project's databases.

1.2.2.13  **Developer**
The Developer role is responsible for designing and implementing the actual solution, based on inputs from the requirements engineering and architecture related roles.
1.2.2.14 Development phase related roles

1.2.2.15 Enterprise architect
The Enterprise Architect is responsible for creating and managing the overall technology infrastructure and ensuring the creation of successful enterprise solutions, which solve a customer's business problems from inception through execution.

1.2.2.16 Higher and middle management related roles

1.2.2.17 IT Director
The IT Director is responsible for proactively managing and implementing the company’s information technology strategy and ensuring the integrity of its processes through long-term strategic goals and short-term tactical plans.

1.2.2.18 IT Manager
The IT Manager is responsible for strategic direction, planning and technical leadership in the organization.

1.2.2.19 Infrastructure architect
The Infrastructure Architect is responsible for developing, communicating, and maintaining the Infrastructure strategies and guiding the implementation of technology platforms needed to implement domain programs and to integrate different domain applications / data.
(Source: http://mike2.openmethodology.org/wiki/Infrastructure_Architect)

1.2.2.20 Network architect
The Network Architect is responsible for overseeing network and data center design, installation, maintenance, and problem resolution. Additionally, the Network Architect installs, configures, and troubleshoots networks and
associated components.

(Source: http://www.epsway.com/company/careers/system-architect/)

1.2.2.21 Operations manager
The operations manager is responsible for overseeing, designing, and redesigning business operations in the production of the goods and/or services of the Organization.

1.2.2.22 Production environment related roles

1.2.2.23 Programmer
see Developer

1.2.2.24 Quality Assurance / Process Manager
QA Manager is responsible for both the resources and tools used to ensure that defects are identified early and do not materialize once the applications go into production.
(Source: http://www.ipsus-it.ie/qa-manager)

1.2.2.25 Release manager
The Release Manager is responsible for planning, scheduling and controlling the movement of releases to test and live environments. His primary objective is to ensure that the integrity of the live environment is protected and that the correct components are released.
(Source: http://wiki.en.it-processmaps.com/index.php/Roles_within_ITIL_V3#Release_Manager)

1.2.2.26 Requirements Engineer
see Business Analyst

1.2.2.27 Requirements engineering phase related roles
1.2.2.28 **Security architect**

The Security Architect is responsible for identifying current and emerging IT vulnerabilities, for championing security standards and processes, review security architecture, and plan, develop and implement IT Security remediation programs.


1.2.2.29 **Software development life-cycle related roles**

1.2.2.30 **System Analyst**

A systems analyst researches problems, plans solutions, recommends software and systems, and coordinates development to meet business or other requirements.

1.2.2.31 **System Architect**

The system architect is the high-level designer of a system to be implemented. The systems architect establishes the basic structure of the system, defining the essential core design features and elements that provide the framework for all that follows, and are the hardest to change later. The systems architect provides the engineering view of the users' vision for what the system needs to be and do, and the paths along which it must be able to evolve, and strives to maintain the integrity of that vision as it evolves during detailed design and implementation.  

1.2.2.32 **System administrator**

The System Administrator is responsible with installing, supporting, and maintaining computer systems, and planning for and responding to service outages and other problems. Additionally his duties may project management for systems-related projects, supervising or training computer operators, and being the consultant for computer problems beyond the knowledge of technical support staff.

(Source: http://en.wikipedia.org/wiki/System_administrator)
1.2.2.33 **Team leader**
A Team Leader is responsible for forming a team with the goal of creating a specific outcome, given set resources and a timescale.

1.2.2.34 **Test Analyst**
A test analyst is responsible for decomposing test requirements, test case creation, test execution and assisting the test lead in defining the test strategy and the test plan.
(Source: http://www.projectrealms.com/testing_realms/glossary_testing_terms_t.htm#test_analyst_role)

1.2.2.35 **Test Developer**
The test developer is responsible for defining, implementing and automating test cases to validate various features.

1.2.2.36 **Test manager**
The test manager helps develop product test strategies, and provides test expertise to the testing group and gathers product information so that corporate management can decide when the product is ready to ship.
(Source: http://www.projectrealms.com/testing_realms/glossary_testing_terms_t.htm#test_manager)

1.2.2.37 **Test phase related roles**

1.2.2.38 **Tester**
A tester is responsible for evaluating the product that has been created by the developer in comparison to the goal defined by the designer. In the end, the software should match the conceptual design as envisioned by the designers, with no logic or performance errors.
(Source: http://www.rocketsoftware.com/u2/products/sbxa/resources/manuals/sb-
1.2.2.39  UI Design

1.2.2.40  UI Designer
  see Usability Engineer

1.2.2.41  Usability Engineer
  The Usability Engineer is responsible for assessing the usability of the application's interface and recommending ways to improve it.

1.2.2.42  User Acceptance Tester
  User Acceptance Tester is responsible for executing user acceptance testing on the final product.

1.2.3  Common Use Cases
1.2.3.1 Automate tasks
Description:
Automate tasks in and across different process and tool boundaries.

1.2.3.2 Collaborate with other users
Description:
Use the builtin collaboration facilities for synchronous/asynchronous communication with other users.

1.2.3.3 Define user roles and permissions
Description:
Define user roles and permissions
1.2.3.4  Filter & Search for information
   Description:
   Filter & Search for information

1.2.3.5  Perform a Check-in/Check-out
   Description:
   Perform a Check-in/Check-out

1.2.3.6  Review other SDLC artifacts (requirements, test cases, change requests, documents)
   Description:
   Review other SDLC artifacts (requirements, test cases, change requests, documents) and use the information as a basis for the system (subsystem, component) design.

1.2.3.7  Share calendars, information, alerts, status, contacts
   Description:
   Share calendars, information, alerts, status, etc.

1.2.4  Primary Use Cases
1.2.4.1 Application portfolio management relevant UCs
1.2.4.1.1 Create and document a road map that outlines and prioritizes overhaul

Description:
Create and document a road map that outlines and prioritizes overhaul

1.2.4.1.2 Define & document enterprise-wide standards, policies and reference models

Description:
Define & document enterprise-wide standards, policies and reference models

1.2.4.1.3 Develop and document a portfolio management and IT strategy

Description:
Develop and document a portfolio management strategy to rationalize, reduce costs and increase
1.2.4.1.4 **Manage the company application portfolio**

**Description:**
Manage the company application portfolio

1.2.4.2 **Customer related UCs**

![Customer related UCs](image)

**Figure 19: Customer related UCs**

1.2.4.2.1 **Access customer view on the projects**

**Description:**
Access a view on the project designed to provide necessary information to external customers
1.2.4.2.2 Communicate and collaborate with project managers, etc.
Description:
Use ALM communication & collaboration system to communicate with project managers, etc.

1.2.4.2.3 Create change request(s)
Description:
Create change request(s)

1.2.4.2.4 Create dashboards and real-time update's relevant to customer
Description:
Create dashboards and real-time update's relevant to customer

1.2.4.2.5 Create project reports
Description:
Create project reports

1.2.4.2.6 Give feedback and escalate problems
Description:
Give feedback and escalate problems

1.2.4.2.7 Participate in review and prototyping sessions

1.2.4.2.8 Receive alerts, review invitations, etc.

1.2.4.2.9 Report problems and seek help
Description:
1.2.4.2.10 **Review real-time project status**

**Description:**
Review real-time project status

1.2.4.3 **Management related UCs**

![Diagram of Management related UCs]

**Figure 20: Management related UCs**

1.2.4.3.1 **Assign requirements to iteration**

**Description:**
Assign requirements to a specific iteration or product version.
1.2.4.3.2 Assign requirements to resources
Description:
Assign requirements to resources.

1.2.4.3.3 Assign tasks to people
Description:
Assign tasks to a responsible person.

1.2.4.3.4 Create project plans and decompose requirements into project tasks and deliverables.
Description:
Create project plans and decompose requirements into project tasks and deliverables.

1.2.4.3.5 Create project reports
Description:
Create project report(s)

1.2.4.3.6 Dokument important decisions
Description:
Dokument important decisions

1.2.4.3.7 Manage product variabilities and product lines
Description:
Manage product variabilities and product lines

1.2.4.3.8 Manage project and business goals
Description:
Manage project and business goals

1.2.4.3.9 Manage project risks
**Description:**
Assess project risks

### 1.2.4.3.10 Organize and manage resources

**Description:**
Organize and manage resources so that they deliver all the work required to complete a project within defined scope, time and cost constraints.

### 1.2.4.3.11 Set and measure key performance indicator

**Description:**
Set and measure key performance indicators, so you can stay on top of the status of projects – as well as each individual task – in real-time.

### 1.2.4.4 Production environment role related UCs

![Diagram of Production environment role related UCs]
1.2.4.4.1  Automate IT operation(s)  
Description:  
Automate IT operation(s)

1.2.4.4.2  Coordinate communication  
Description:  
Coordinate communication

1.2.4.4.3  Create and document a service strategy  
Description:  
Create and document a service strategy

1.2.4.4.4  Create and use a Service Desk  
Description:  
Create and use a Service Desk

1.2.4.4.5  Create product trainings and make it available to the product users  
Description:  
Create product trainings and make it available to the product users

1.2.4.4.6  Define and assess measures for IT Services  
Description:  
Define and assess measures for IT Services

1.2.4.4.7  Design and test IT Services  
Description:  
Design and test IT Services

1.2.4.4.8  Document and manage Product Knowledge
1.2.4.4.9 **Performing first-line product diagnosis**

Description:
Performing first-line diagnosis

1.2.4.4.10 **View call and problem logs**

Description:
View call and problem logs

1.2.4.5 **SDCL role related UCs**

![Figure 22: SDCL role related UCs](image_url)

1.2.4.5.1 **Architecture role related UCs**
1.2.4.5.1.1 Assign tasks to developers
Description:
Assign tasks (requirements, components, etc.) to developers for completion.

1.2.4.5.1.2 Create & maintain traceability
Description:
Maintain traceability between requirements and architectural components and decisions.

1.2.4.5.1.3 Create architectural models
Description:
Create architectural (software, system, network and/or database) models to
abstracts the complexity of a system by decomposing it into manageable models

1.2.4.5.1.4    Create architectural specifications
   Description:
   Create architectural documents (e.g. specification, presentations, etc.) from the architectural model & documented architectural descriptions.

1.2.4.5.1.5    Document architectural decisions
   Description:
   Document all relevant architectural decisions.

1.2.4.5.1.6    Oversee and manage the development process
   Description:
   Define deliverables and create timeline by which the system must be designed and continuously manage and oversee the development process.

1.2.4.5.1.7    Understand market trends
   Description:
   Review vision & scope and other market analysis documents to understand the market trends and the organizational goals behind the product / system development

1.2.4.5.1.8    Understand the stakeholder's needs and constraints
   Description:
   Review domain model and requirements artefacts to understand the domain and the stakeholder's needs and constraints

1.2.4.5.2    Configuration management role related UCs
1.2.4.5.2.1  Build system

Description:
Compile and link the software components into an executable system.

1.2.4.5.2.2  Create & document a CM plan and assign responsibilities

Description:
Defines the types of documents to be managed, their naming scheme, the responsibility for the CM procedures and creation of baselines.

1.2.4.5.2.3  Create and document procedures and standards for evolving software

Description:
Development and document procedures and standards to manage change control and version management of an evolving software product.
1.2.4.5.2.4 Create, manage branch or baseline
   Description:
   Create, manage (maintenance) branches and baselines (of requirements, test cases, issues, code, models, test data, user manuals, specifications, etc.)

1.2.4.5.2.5 Display derivation history
   Description:
   Display records of made changes, the rationale for the change, who made the change and when it was implemented.

1.2.4.5.2.6 Enforce the CM procedures and standards (naming schemes, etc.)
   Description:
   Enforce the created procedures and standards (naming schemes, etc.) in the entire system

1.2.4.5.2.7 Identify the current combinations of components.
   Description:
   Identify the current combinations of components.

1.2.4.5.2.8 Plan and distribute new system releases
   Description:
   Plan and distribute new system releases

1.2.4.5.3 Development role related UCs
1.2.4.5.3.1 **Allocate Requirements to Design Artifacts.**

**Description:**
Allocate the requirements to the design artefacts for the purpose of traceability.

1.2.4.5.3.2 **Allocate time to task**

**Description:**
Allocate the time spend on a specific task in the timesheet functionality.

1.2.4.5.3.3 **Automatically derive / update project metrics based on the progress report**

**Description:**
Derive / Update project metrics based on the progress report.

1.2.4.5.3.4 **Create progress report on assigned tasks**

**Description:**
Report progress on assigned tasks (requirements, components, etc.)
1.2.4.5.3.5 Create system design  
**Description:**  
Using the requirements as a basis, create a design for the assigned component, etc.

1.2.4.5.3.6 Implement required functionality  
**Description:**  
Use builtin code-editing (IDE) capabilities to implement the requested functionality / feature/ requirements, etc.

1.2.4.5.4 Process related UCs

![Diagram of Process related UCs](image)

**Figure 26:** Process related UCs

1.2.4.5.4.1 Customize the system according to the chosen development methodology  
**Description:**  
Customize the system according to the chosen development methodology

1.2.4.5.4.2 Customize the system according to the chosen development methodology  
**Description:**
Customize the system to the chosen development methodology (e.g. agile, waterfall, etc.).

### 1.2.4.5.5 RE role related UCs

**Figure 27: RE role related UCs**

#### 1.2.4.5.5.1 Create baselines

**Description:**
Create baselines of the requirements engineering relevant artifacts

#### 1.2.4.5.5.2 Create goal model

**Description:**
Create goal models
1.2.4.5.3 Create update traces  
Description: 
Create / Update traces to other relevant artefact (test cases, risks, tasks, architecture & design artefacts, code, etc.)

1.2.4.5.4 Create variability model  
Description: 
Create variability models

1.2.4.5.5 Elicit, define and manage requirements-relevant information  
Description: 
Elicit, define & manage stakeholder requests, requirements, features, use cases, risks, mitigations, goal models, etc.

1.2.4.5.6 Generate Specifications  
Description: 
Automatically create requirements specifications

1.2.4.5.7 Perform impact analysis after change requests  
Description: 
Perform impact analysis after change requests

1.2.4.5.8 Propagate (changed, new) status and priorities to other artifacts  
Description: 
Propagate (changed, new) status and priorities to other artifacts

1.2.4.5.6 Test related UCs
1.2.4.5.6.1  Assign resources (tester, time frame, etc.) to test case

Description:
Assign resources (tester, time frame, etc.) to test case

1.2.4.5.6.2  Automatically create bug reports and tasks for developers based on test failures

Description:
Automatically create bug reports and tasks for developers based on test failures

1.2.4.5.6.3  Automatically run test script runs and report results

Description:
Automatically run test script runs and report results (e.g. run regression suites on change in requirements, etc.)

1.2.4.5.6.4  Create & manage tests and link them to their corresponding work items, requirements, etc.

Description:
Create tests and link them to their corresponding work items such as requirements, change requests, other test cases, etc.
1.2.4.5.6.5 Create regression test suite based on different criteria and/or criteria combinations

Description:
Create regression test suite based on different criteria and/or criteria combinations (risk, time, severity, impact analysis results, etc.).

1.2.4.5.6.6 Create, Plan & run tests

Description:
Plan tests, create test suites and test configurations based on different criteria & run tests.

1.2.4.5.6.7 Generate test case(s) from UML model

Description:
Generate test case(s) from UML model(s).

1.2.4.5.6.8 Outline and document quality assurance procedures

Description:
Outlines and document all quality assurance procedures.

1.2.4.5.6.9 Report and/or track testing progress and status of bugs

Description:
Report and/or track testing progress and status of bugs

1.2.4.5.6.10 Submit bugs

Description:
Submit a bug.

1.2.4.5.7 UI design role related UCs
1.2.4.5.7.1 Create a cost-effort estimate for a design option

Description:
Create a cost-effort estimate for a design option.

1.2.4.5.7.2 Create a storyboard

Description:
Create a storyboard to illustrate specific interaction.

1.2.4.5.7.3 Create a workflow model

Description:
Create a workflow model describing how the system fits in with the user's normal workflow or daily activities.

1.2.4.5.7.4 Create and document UI Design

Description:
Create and document an UI Design

1.2.4.5.7.5 Trace UI components and changes to UI components to other artefacts

**Description:**
Create and maintain traceability between UI components and other engineering artefacts.

1.2.4.5.8 Variability Management

**Figure 30: Variability Management**

1.2.4.5.8.1 Define and document a strategic for systematic reuse

**Description:**
Define and document a strategic for systematic reuse
1.2.4.5.8.2 Define and manage application variability

1.2.4.5.8.3 Identify and manage common and variable assets
   
   Description:
   Identify and manage common and variable assets
Appendix F
ALM 2.0 - Design concepts & ideas

Integration means
- RESTful Web services
- Metamodel-level Integration
- DSL-based Integration
- Multi-level orthogonal integration
- Service & Control level
- Ontology & Data level
- Method & Process level
- Enterprise Bus Integration
- SOA-based Integration
- Ontology-based semantic Integration
- AOP-based Integration

Getting commitment
- European Funded project
- Joint Venture

Grand Challenges & Research Needs
- Better understanding for tool integrations
- kinds of data to be integrated
- how is the data used
- Use of workflow systems
- Unite contradicting expectations
- Migration of old monolithic tool architectures towards an integration open architecture
- Semantic Standard
- Solve the syntactic and control issues
- Integrate web-based tools
- Integrate web-based tools