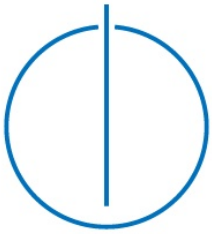
Fakultät für Informatik

der Technischen Universität München

Bachelorarbeit in Wirtschaftsinformatik

Roadmaps for Enterprise Architecture Evolution

Alexander Schneider



Fakultät für Informatik

der Technischen Universität München

Bachelorarbeit in Wirtschaftsinformatik

Roadmaps for Enterprise Architecture Evolution

Roadmaps für die Entwicklung von Unternehmensarchitekturen

Author:	Alexander Schneider
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	Alexander Ernst
Submission Date:	15.09.2009

I assure the single handed composition of this bachelor's thesis only supported by declared resources.

Munich, 15.09.2009

(Alexander Schneider)

Abstract

Enterprise Architecture (EA) management is an emerging and hence constantly evolving discipline which targets at establishing methods for aligning business and information technology (IT) on a company-wide level. For example, the EA describes the interaction of applications, business processes, and organizational units. Managing the evolution of the Enterprise Architecture is a key challenge for modern enterprises. Application landscapes are growing uncontrolled and without management complexity and maintenance costs increase. The first step of an EA approach is to analyze and document the as-is-situation. Subsequently, the second step has to be the management of the EA evolution. Although IT projects guide this evolution by implementing the transformations from a current to an envisioned architecture, they are considered sparsely by EA approaches. Therefore, this thesis determines a process and roles needed to manage software development projects in a portfolio context, regarding their effective definition and their interrelationships. Furthermore, viewpoints for a roadmap and for the migration of business supports are provided. To automatically generate these visualizations and to show which information is needed during the mentioned process, information models are also supplied. The documentation follows the pattern-based approach to EA management also used during the development of the *EAM Pattern Catalog*. In order to document proven practice solutions, this thesis is supported by a German reinsurance company. Their current approach to EA evolution management is analyzed and complemented by additional requirements surveyed by interviews. These interviews aim to cover a broad range of stakeholders and users and are conducted with people from several divisions and hierarchy levels. Finally, the newly documented EAM Patterns are linked to existing patterns of the *EAM Pattern Catalog* and the benefits and liabilities of the pattern-based approach to EA management will be assessed. Topics for future research in EA evolution management conclude this thesis.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Enterprise Architecture (EA) management has become an important topic in academia and practice over the last years. Due to business processes becoming more differentiated and complex, more requests for additional applications emerge in nearly every company relying on competitive information technology (IT) advantages. As a result, the application landscape grows uncontrolled, becomes unmanageable, and needs to be consolidated. There also exist legal requirements, e.g. Sarbanes-Oxley Act [SOX02], which forces companies to manage their EA. After implementing an EA management approach, they are especially faced with the challenge to manage the evolution of their EA. There are several reasons why an EA will change. For example, new business processes require new applications or technical reasons make changes to applications necessary. The major challenge is to handle the high complexity of parallel new application introductions, business support migrations, and consistent retirement of legacy systems. Therefore, it is not enough to document inter-application dependencies occurring during their runtime. Furthermore, dependencies during the application development (inter-project dependencies) must also be taken into account, to ensure the availability of developed applications and their business supports on schedule. In order to support EA management during this difficult task of EA evolution management, appropriate processes, performing roles, visualizations, and information models are needed.

1.2 Designated targets

In this thesis the current approach to manage the evolution of the EA implemented at a German reinsurance company should be documented and improved. Therefore, the current approach has to be analyzed and partly reverse engineered. In addition, a respective literature analysis covering relevant disciplines has to be performed. These disciplines are without limitation EA management, project management, multi-project management, the critical path method, release management, and risk management. In order to improve the current approach and to satisfy stakeholders and users, their requirements have to be determined and evaluated. Therefore, interviews will be conducted with people from several divisions and hierarchy levels. The documentation of the enhanced approach to EA evolution management should follow the pattern-based approach to EA management, also used in the *EAM Pattern Catalog*. The documented approach should provide a process and performing roles for the management of EA evolution. Furthermore, appropriate visualizations have to be determined. To complete the approach, information models showing the information required during the process and for visualization generation should also be provided. The documented EAM Patterns extend the *Enterprise Architecture Management (EAM) Pattern Catalog* [seb09a]. They are not tied to the company supporting this thesis and can be adopted by each company to create their own company specific roadmap for Enterprise Architecture evolution.

1.3 Environment Description

This section will describe the environment in which this thesis takes place. First, the encompassing research project is introduced. Second, the company supporting this thesis is shortly presented.

1.3.1 System Cartography Project

The System Cartography Project is a research project of the Chair for Informatics 19 (sebis) at the Technische Universität München. The project objective is to determine "which concepts and notations are able to show the economical and managerial meaning of information systems [...] in an

understandable way" [seb09b]. Therefore, many large German companies are involved in this project. As one part of the project, the *SyCaTool* is developed, which should describe, evaluate, and visualize application landscapes in order to get transparency. Another part is the *EAM Pattern Catalog* which will be described in the next section. This thesis is within the scope of the System Cartography Project because the resulting patterns might be included in the *EAM Pattern Catalog* after their validation.

EAM Pattern Catalog

"*The objective of the EAM Pattern Catalog is to complement existing Enterprise Architecture (EA) management frameworks, which provide a holistic and generic view on the problem of EA management, and to provide additional detail and guidance needed to systematically establish EA management in a step-wise fashion within an enterprise*" [seb09a].

In general, the patterns of the *EAM Pattern Catalog* describe proven practice solutions for recurring problems that can and may have to be adapted to a specific enterprise context. They can be subdivided into four types of patterns. Methodology patterns (M-Patterns) describe, which activities have to be performed or which roles have to be assigned, in order to solve reoccurring problems. The second type of EAM patterns are viewpoint patterns (V-Pattern). They show, which diagrams, figures, tables, and listings help stakeholders to collaboratively perform these activities mentioned in the related M-Patterns. The third type consists of information model patterns (I-Pattern), which describe the information required to create a particular viewpoint. The fourth type are Anti-Patterns. These EAM patterns describe approaches which turned out not to solve the addressed problem. They also describe lessons learned and are often used if there is no pattern of one of the other three types available yet. In order to get an overview of relations between individual patterns, a pattern map shows all relations between the different patterns and groups according to their type [seb09a]. In addition, the pattern map can be used to select patterns related to each other. For example, if there are several viewpoints addressing the same problem, the pattern map provides an easy way to find these V-Patterns and then select the appropriate. Figure 1.1 shows a snippet of the *EAM Pattern Catalog* pattern map.

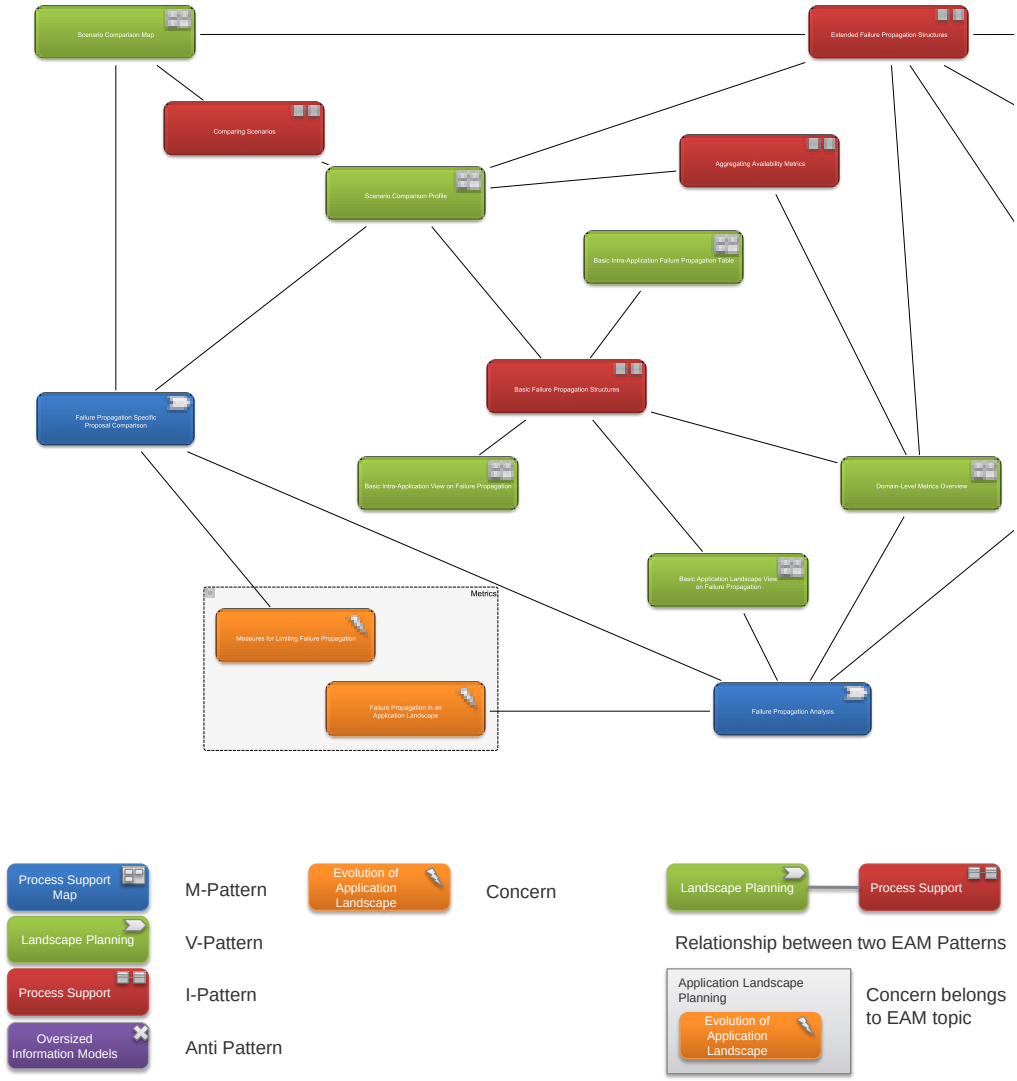


Figure 1.1: Exemplary snippet of the *EAM Pattern Catalog* pattern map

1.3.2 Munich Re

Munich Re is the world market leading reinsurance company and has about 5,000 customers. It was founded in 1880 at the instigation of Carl von Thieme, Baron Theodor von Cramer-Klett, and Wilhelm Finck. Its core business is to analyze and bear high complex risks, which requires expertise in many disciplines [Mun09]. The Munich Re Group consists of three major business branches, namely *Reinsurance*, *Munich Health*, and *Primary Insurance*. In general, the reinsurance branch insures so called primary insurance companies as well as other reinsurance companies. Therefore,

Munich Re takes over parts of the risk of live, health, liability, casualty, aerospace, fire, and other insurances. Munich Health is a relatively new individual branch, which combines primary health insurance companies and reinsurance departments responsible for health insurance. The primary insurance branch, also known as ERGO, contains several primary insurance companies. In fact, these companies are just brands and all use the same business processes. In addition, a company called Munich Ergo Assetmanagement GmbH (MEAG) is responsible for asset management across all branches.

Figure 1.2: Global sites of Munich Re

With over 40.000 employees Munich Re Group is present on every continent which can be seen in Figure 1.2. Due to the fact that the reinsurance business, with all of its financial and risk management facets, depends intensively on supporting IT, Munich Re also employs about 500 people in its IT departments worldwide. The main reason for keeping its own IT department is that IT support is not only a need, but especially core-business supporting applications are expected to create competitive advantages. Therefore, high experienced people with knowledge in reinsurance business and IT are essential. Other reasons are keeping knowledge within the company and the ability to control outsourcing activities. In addition, reinsurance can be considered as a niche-business. That means, that there are relatively few competitors. As a consequence, IT professionals specialized for reinsurance business are rare on the market. The last reason why Munich Re keeps its own IT department is that only internal specialists can evaluate new IT trends and determine which can improve IT support.

The EA management approach of Munich Re is developed and implemented since 2000 and constantly gains more importance. The focus is on business-IT-alignment, and therefore, mainly on *governance*, *business architecture*, and *information architecture*. The technical architecture is not in scope at the moment. What Munich Re wants to achieve by their EA management according to their public EA management presentation [Wal09] is:

- Make business architecture more comprehensible
- Reduce complexity of business and the resulting impact on IT
- Identify at an early stage, how changes in business affect IT
- Establish *IT enables business* instead of *IT follows business*
- Eliminate redundancies within the application portfolio
- Identify white spots/red spots of the Enterprise Architecture
- Adapt IT to changes in technology in a more controlled manner

Munich Re strictly acts according to their postulate *restrict EAM to those data you need to answer your architectural questions/concerns*. That means that no data will be collected or evaluated, which does not directly answer architectural questions and no out-of-the-box approach or tool will be used. These architectural questions include without limitation following questions [Wal09]:

- Which processes are currently/will be later supported for which organization by which applications?
- Which applications (versions) will be available when?
- Which applications are global/local?
- What are the necessary business functions to perform our business?

In order to answer these questions, Munich Re maps all applications worldwide to the process they support and to the organizational unit they were used at. In addition, the location where the application is physically running is also documented. Furthermore, all interfaces between applications are determined. With these information, the global business architecture (GBA) department and IT strategy are able to evaluate project proposals

respecting their impacts to the application landscape and therefore to the EA. Without the approval from GBA, which is responsible for EA management, no project can start anymore. The evaluation of project proposals might be a difficult task because many projects transform the EA at the same time. This leads to the need for a roadmap visualizing rollout dates and project dependencies for at least the next two years. Such a roadmap would help people to determine all dependencies and impacts of an additional project and allows them even to do resource planning on a high level. In addition, a roadmap can be used as management view. Higher-level managers are often interested in overviews to assume complexity. Another reason is the ability to do manual quality assurance for all projects and their dependencies. Finally, it can be used as a basis for discussions about the validity of the data that is shown.

1.4 Approach

The first chapter includes the motivation for this thesis and defines the designated targets. Additionally, it describes the environment in which this thesis takes place. The first step of the approach which should reach the targets defined in Chapter 1.2, is a literature analysis of related disciplines which is described in Chapter 2.1. The fields of knowledge which are regarded to be of interest are EA management, project management, multi-project management, the critical path method, release management, and risk management. Moreover, the current approach of Munich Re will be analyzed, regarding its stakeholders, viewpoints, and information. Both, the related work and the analysis of the current approach are a prerequisite to the requirements elicitation done in Chapter 3 because the interview guideline leading through the respective interviews is developed regarding this knowledge. In order to determine the requirements for a roadmap for enterprise architecture evolution, interviews with stakeholders from different divisions will be conducted using the before mentioned interview guideline. In Chapter 4 the insights of the other chapters are documented as EAM Patterns using the pattern-based approach to EA management. This includes a methodology pattern, viewpoint patterns for possible visualizations, and information model patterns describing the necessary data. Chapter 5 concludes this thesis by a review and an assessment of the used pattern-based approach to EA management and provides an outlook of possible further research topics.

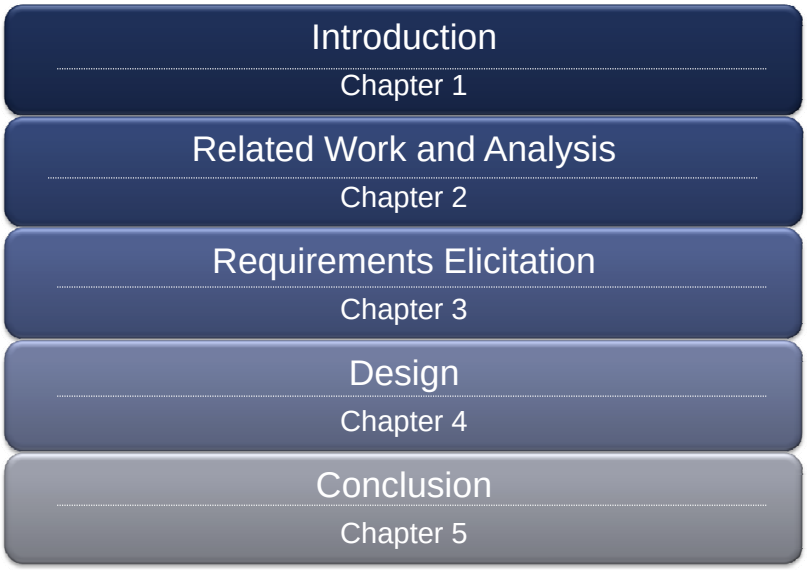


Figure 1.3: Structure of this thesis

Chapter 2

State of the art in Enterprise Architecture Management

This chapter gives an overview about the state of the art in EA management and related disciplines, for example project management and multi-project management which are relevant for a roadmap for enterprise architecture evolution. Subsequently, Munich Re's current approach to a roadmap for enterprise architecture evolution is analyzed.

2.1 Enterprise Architecture Evolution in Literature and Practice

For the evolution of the enterprise architecture several disciplines must be taken into account. These are without limitation: project management, multi-project management, the critical path method, release management, and risk management. Each of these disciplines is shortly described in this section and the respective impacts to the roadmap are depicted.

2.1.1 Enterprise Architecture Management

The term enterprise architecture (EA) is pervasive and accepted in science and practice. Nevertheless, there exist many definitions and understandings of an EA. A good general definition of architecture can be found in the IEEE standard 1471-2000 [IEE07]. It says architecture is

> the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution

which applies also to EAs. Since the 1980s, when the discussion about the description and the development of EA began, many frameworks appeared in literature. The best-known are the *Zachman framework* [Zac99] and *The Open Group Enterprise Architecture Framework* (TOGAF) [The09]. According to Winter, Riege and Aier [ARW08], it is hard to compare them, due to their complexity and their differences in parts of their objectives. Therefore, they identified artifacts of an EA and derived architecture layers, shown in Figure 2.1, from a previous literature analysis.

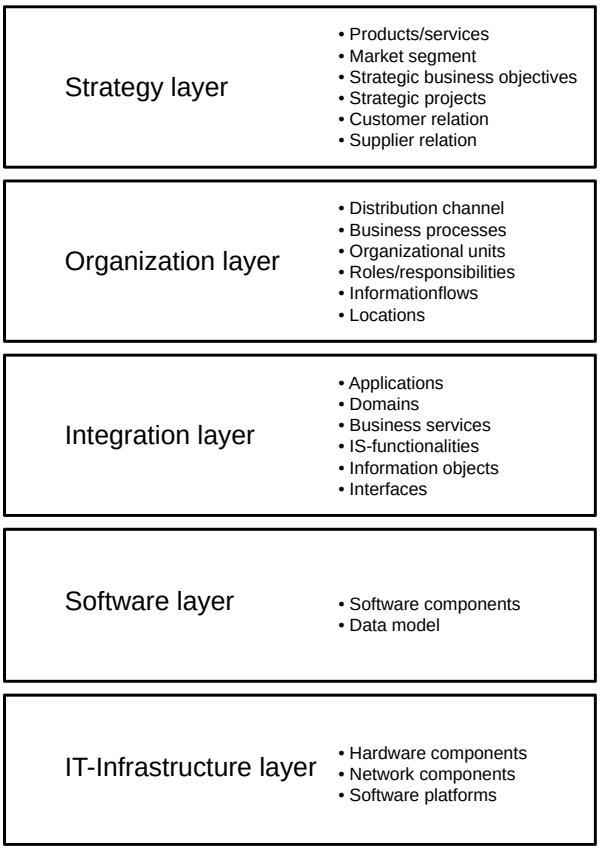


Figure 2.1: Layers of EA according to Winter

The objective of EA-models is to show dependencies of included objects on an aggregated level. On the one hand, this can be done for the as-is-state of the EA, for documentation or analysis reasons. On the other hand, it can be done for the target-state, for planning reasons. The objects that can be included in EA-models, are listed in Figure 2.1.

Like other companies, Munich Re has its own definition for EA: *An enterprise architecture is a conceptual blueprint that defines the structure and operation of an organization. The intent of an enterprise architecture is to determine how an organization can most effectively achieve its current and future objectives.*

According to [LW04], there is no established definition of EA management. Here, the definition employed in the *Enterprise Architecture Management Tool Survey 2005* [seb05] is used:

"EA management is a continuous and iterative process controlling and improving the existing and planned IT support for an organization. The process not only considers the information technology (IT) of the enterprise, but also business processes, business goals, strategies, etc. are considered in order to build a holistic and integrated view on the enterprise. The goal is a common vision regarding the status quo of business and IT as well as of opportunities and problems arising from these fields, used as a basis for a continually aligned steering of IT and business."

As an example how companies manage their EA, see Section 1.3.2.

2.1.2 Project Management

Projects are complex endeavors and project outcomes are far from being certain [DLP02]. Project management owes its existence as a management discipline to the complex, high technology undertakings like developments in space programs and nuclear arms [Mor97]. Even though the modern project management discipline has been around for almost sixty years delivering successful technology projects is still an obstacle for many organizations. The Standish Group reports in their *Chaos Study* that in 2000 only 28% of all IT application development projects have succeeded, while all others were canceled before completion or did not achieve the project goals like time, cost, and specifications [Joh01]. Over the last 50 years *cost*, *time*, and *quality* (The Iron Triangle) have become inextricably linked with measuring the success of project management and are often included in the definition of project management. An early attempt was made by Oison [Ois71]:

> Project Management is the application of a collection of tools and techniques [...] to direct the use of diverse resources

> toward the accomplishment of a unique, complex, one-time task within time, cost and quality constraints. Each task requires a particular mix of theses tools and techniques structured to fit the task environment and life cycle (from conception to completion) of the task.

But even newer definitions of projects and project management still include these three success criteria of the Iron Triangle. For example, the UK Association of Project Management (APM) has produced a UK Body of Knowledge [Dun96] which also provides a definition for project management as:

> The planning, organization, monitoring and control of all aspects of a project and the motivation of all involved to achieve the project objectives safely and within agreed time, cost and performance criteria. The project manager is the single point of responsibility for achieving this.

Turner [Tur96] further suggested that project management could be described as:

> The art and science of converting vision into reality.

This definition does not include any success criteria or measures and is vague. None of the mentioned definitions describes the concrete process and tasks of project management. As a conclusion, project management is too wide a subject to include all facets in a definition. But, of course literature provides guidelines and defines activities to realize a good project management in order to achieve project success.

A major issue of project management is to deal with project complexity [BD04]. The term project complexity emerged during the 1980s and 90s in project management history [Bac96]. Fioretti and Visser [FV04] define complexity in terms of *inadequacy knowledge needed to solve a problem*. This complexity needs to be reduced as much as possible by adequate project management techniques. For example, these tasks of project management are problem statement creation, founding a project organization, assigning roles, planning which includes a work break down structure, estimation and scheduling, and dealing with risks [BD04].

The project management discipline has direct impact on a roadmap for enterprise architecture evolution. Projects are not considered to be managed by the roadmap but it can support project managers during their planning activity. For example, the roadmap provides information about specific delivery dates and available resources might be derived. Furthermore, the basic part of data needed for and included in the roadmap originates from project plans made by project managers. This requires standardized project plans in order to allow automation. Because the roadmap is for the main part used for project management, the roadmap should be oriented towards established forms of presentation and use familiar terms.

There are different methodologies for project management. The traditional way to manage projects is to create a project plan according to the given budget and deadline. There are standards for such a plan like the Software Project Management Plan by IEEE [IEE98]. The common behavior in traditional project management is, if any deviation from the plan occurs, that the project manager tries to get back to the plan. A relatively new developed methodology is called agile project management. According to the *Manifesto for Agile Software Development* [BBB+01], agile project management focuses on individuals and interactions over processes and tools, working software over comprehensive documentation, and responding to change over following a plan. Of course, even agile project management uses some kind of plans, but they are likely to change more frequently. Therefore, an overall roadmap how it is considered in this thesis is only sensible for the traditional way of project management, not for the agile.

2.1.3 Multi-Project Management

Modern project-driven organizations conduct many software development projects in parallel, each managed by an individual project manager. Often resources, for example people, hardware, or data are needed by more than one project at the same time. This fact raises the demand for an overall management of projects. This activity is called multi-project management, which can be divided into two sub-activities namely *program management* and *project portfolio management* (PPM). A program management approach is appropriate, if a coherent group of single projects is managed in a coordinated way to add benefit [Tur04]. Murray-Webster and Thiry [MWT00] define a program as a collection of projects purposefully grouped together to realize strategic and tactical benefits. Using a portfolio approach means that decisions are based on the full set of company projects.

They all target their individual goals, but have to align to the company's strategy. Instead of focusing on the challenge "doing projects right" which is done by project management, PPM focuses also on "doing the right projects" [RGCL+05]. The field of project portfolio management owes its origins to a seminal paper written by Harry Markowitz [Mar52] in 1952 in which he laid down the basis for the *Modern Portfolio Theory*. It was initially developed for financial investments, but in 1981 McFarlan [McF89] provided the basis for the modern discipline of PPM for IT projects.

For a roadmap regarding the EA evolution, a PPM approach should be chosen because all projects of a company are of interest and not only coherent groups. In [RGCL+05] key elements of PPM are identified. These are: *centralized view of the project portfolio, risk analysis, interdependencies, prioritization, alignment*, and *selection*. According to Keller [Kel07], a PPM approach also helps to ensure project conformity to IT strategy, reduction of complexity, reduction of risks, and avoid redundancy.

The project portfolio management discipline has direct impact on a roadmap for enterprise architecture evolution. The roadmap is considered to show dependencies between projects and support managers at their risk management and prioritization activity. In addition, if planned and proposed projects are also included, the roadmap can serve as a basis for selecting compatible projects. It also provides a centralized overview of at least the most important projects.

2.1.4 Critical Path Method

During the late 1950s the critical path method (CPM) was developed through efforts of DuPont and Remington Rand to create a formalized project management tool. It is a network technique used to help in the planning, scheduling, monitoring, and control of activities which are related to each other. By contrast to other techniques, for example, the program evaluation and review technique (PERT), CPM treats activity performance times in a deterministic manner, so it assumes that the activity times are known with certainty [SW69]. As a basis, the CPM uses a project network diagram. This is a graph representation of activity performance times and dependencies between activities. According to [Luc05] there exist two different types of network diagrams, namely the activity-on-the-arrow (AOA) diagram and the activity-on-the-node (AON) diagram. They only differ in the graphical notation. AOA employs arrows to symbolize the dynamic

nature of the activities and nodes to show their connections, while AON uses rectangles or circles as the body of the activities and arrows for the logic associations between them. In order to obtain such a directional network diagram, the first step is to determine a list of all activities, which need to be done to reach the project goal, together with their initial and terminal events. Then for each activity, predecessor and successor relations need to be identified. With this data, the network diagram can be drawn. Another rendering possibility to the notation as a graph is a Gantt chart, which displays also the duration of activities in a graphical way by using bars. Moving to an activity in the graph, in order to initiate it, is only possible after each predecessor activity has been completed. At this point in time, the planning (list of activities) and the scheduling (relations) are done. To use the monitoring and control functions of the CPM, the *critical path* needs to be found. The critical path is a sequence of activities which are critical to the completion of the project in that if any activity on this path is delayed, the completion of the project is delayed. Therefore, the *forward-backward algorithm* [SW69] is used to determine the earliest and the latest start and end time for each activity. The critical path consists of those activities, for which earliest start time and latest start time are equal. The difference between the earliest and the latest start time is called the total slack for that activity. Because activities on the critical path have a slack of zero, there is no flexibility for rescheduling and as mentioned before, delays have direct impact on the total duration of the project.

Figure 2.2 shows an example of a small project plan and the highlighted critical path. It only shows the earliest start and the latest end time. The latest start and earliest end time are excluded. The diagram visualizes that activities A, E, and G are not on the critical path of that project. These activities have slack times, which means that a delay of that activity which is not longer than the specific slack time, does not affect the duration of the project. All other activities are on the critical path and so every delay will result in an extension of the project duration.

For the roadmap, the CPM can be used to determine the critical path for all projects included. This might help to prioritize projects in order to complete them as initially planned. Projects on the critical path need special attention because they cannot bear delays. The CPM is implemented by many project management tools, like Microsoft Project, so there is no need to compute the critical path by hand.

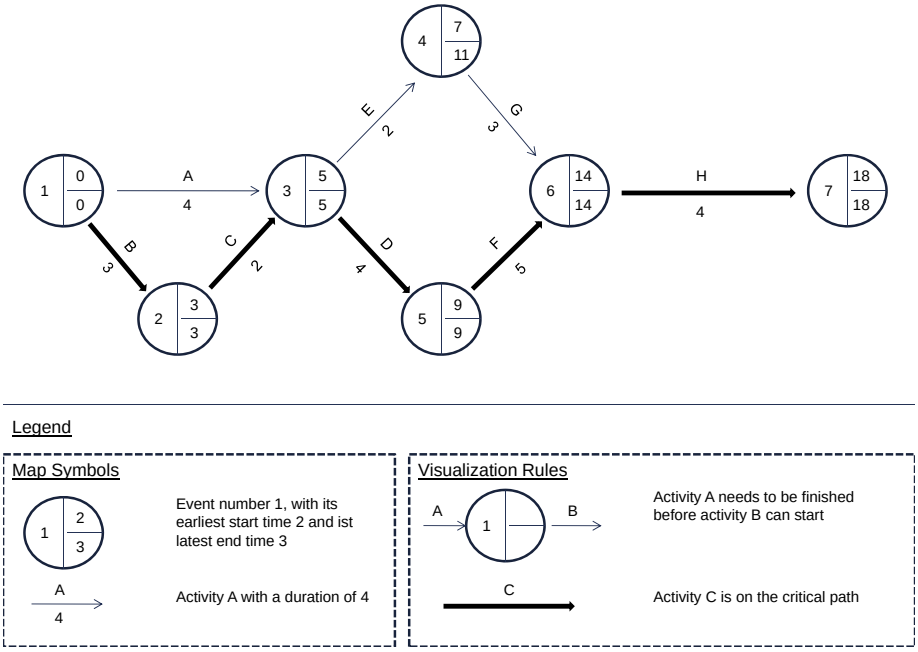


Figure 2.2: Example of a project plan with a critical path [Cou09]

2.1.5 Release Management

In the past a single organization developed a software system on its own. At this time, software configuration management systems (e.g. ClearCase and Continuus) were used to support release management. In case of a release, all components were frozen and users were informed about the availability of the new release. Since the advent of the Internet and distributed development, developers and users required multiple distribution channels and less needed effort [HHHW97]. Additionally, developers want to control the scope of a release in order to show it only to selected users and keep a history of retrievals. By contrast, users request for detailed release descriptions, a hidden physical distribution, and independent systems retrievable as one. All these requests need to be addressed by an appropriate release management and can not be achieved only by configuration management.

According to Hoek et al. [HHHW97] release management is the process through which software is made available to and obtained by its users. Rana and Arfi [RA05] consider this process to be the key success factor of any software product. The definition found in [CA09] defines release management as:

> The process of planning, building, testing and deploying hardware and software, the version control and storage of software.

But this process includes several challenges. For example, the developers have to document accurately high complex and changing dependencies between components. Furthermore, today's software is often developed in a distributed and decentralized way and components can be developed and released individually. The user is also involved in the release process. He has to retrieve a consistent set of components from multiple sources, possibly via multiple methods, and place it within the context of his local environment [HHHW97].

Another field of the release management discipline is called release planning. According to [GR04], there is an increasing tendency to develop and deliver software in an incremental fashion, in order to achieve higher flexibility and to better satisfy actual customer requirements. This means, instead of delivering a monolithic system after a long development time, smaller releases are implemented and delivered sequentially. Each release is a completed product useful to the customer. But some challenges need to be accepted. Delivering software incrementally necessitates a process of requirements prioritization and assigning them to releases. Due to different stakeholder perspectives, especially requirements prioritization might be a challenging task. Greer and Ruhe [GR04] name three main considerations that need to be taken into account. These are the *technical precedence* inherent in the requirements, the typically *conflicting priorities* as determined by the representative stakeholders, as well as the *balance between required and available effort*. In summary, incremental software development is not straightforward to perform and requires some challenging activities. But the result will be a shorter time-to-delivery for high priority features and higher customer satisfaction due to the possibility of feedback after each release.

The release management discipline has direct impact on a roadmap for enterprise architecture evolution. The way of delivery has to be taken into account in rollout planning because it might cause dependencies. If software is developed and released incrementally, more releases and also dependencies between them need to be managed by the roadmap which results in higher complexity.

2.1.6 Risk Management

> "One aspect of the future is obvious: all new undertakings will be accomplished in an increasingly complex technical, economic, political and social environment. Thus project management must learn to deal with a much broader range of issues, requirements, and problems in directing their projects to successful conclusions. Certainly, project management in every field will be called upon to address complexities and risks beyond anything experienced in the past" [Tum86].

Tuman states that risk management is one of several other activities needed to be performed in project management. A *risk* is mostly considered as an uncertain event which might have impact to the project. Williams [Wil95] summarizes several scales which are used in literature to classify a risk, namely *impact, likelihood*, and *predictability*. In order to manage risks, several tasks need to be performed. According to the National Institute of Standards and Technology [SGF01] these tasks are:

- System Characterization
- Vulnerability Identification
- Threat Identification
- Control Analysis
- Likelihood Determination
- Impact Analysis
- Risk Determination
- Control Recommendations
- Results Documentation

The result documentation is often done with a so called risk matrix. There, every risk is classified according to its impact and its likelihood. An example of such a matrix is shown in Figure 2.3. As a result, everybody can see immediately, if a risk is *low* (green, L), *medium* (yellow, M), *high* (orange, H) or *extreme* (red, E).

Likelihood	Consequences				
	Insignificant	Minor	Moderate	Major	Severe
Almost certain	M	H	H	E	E
Likely	M	M	H	H	E
Possible	L	M	M	H	E
Unlikely	L	M	M	M	H
Rare	L	L	M	M	H

Figure 2.3: Example of a risk matrix [AG09]

The risk management discipline has direct impact on a roadmap for enterprise architecture evolution. The roadmap and the provided information can be used to support the risk management activity. For example, the process creating the roadmap ensures a high level of information about the project portfolio, which makes risk identification easier. The determination of all external sources a project depends on helps to identify risk sources. Furthermore, an impact analysis can deliver better results, if inter-project dependencies visualized in the roadmap for the whole project portfolio are also considered.

2.2 Analysis of Munich Re's current approach

In this section, the reinsurance's current approach to manage inter-project dependencies and application rollouts will be depicted. The current approach is a Microsoft Excel document containing several sheets with different views. A snippet of the main view of this current approach is shown in Figure 2.4. First, the focus of the analysis will be on the stakeholders of this approach. The identification of stakeholders will be used as a basis for determining interview partners for the requirements elicitation depicted in Chapter 3. Second, the way of its presentation will be analyzed to ascertain which views are relevant, which information is shown, and which visualization rules are applied. Third, the implicit underlying information model will be reverse engineered because an information model is essential, if views should be generated by tools and not by hand.

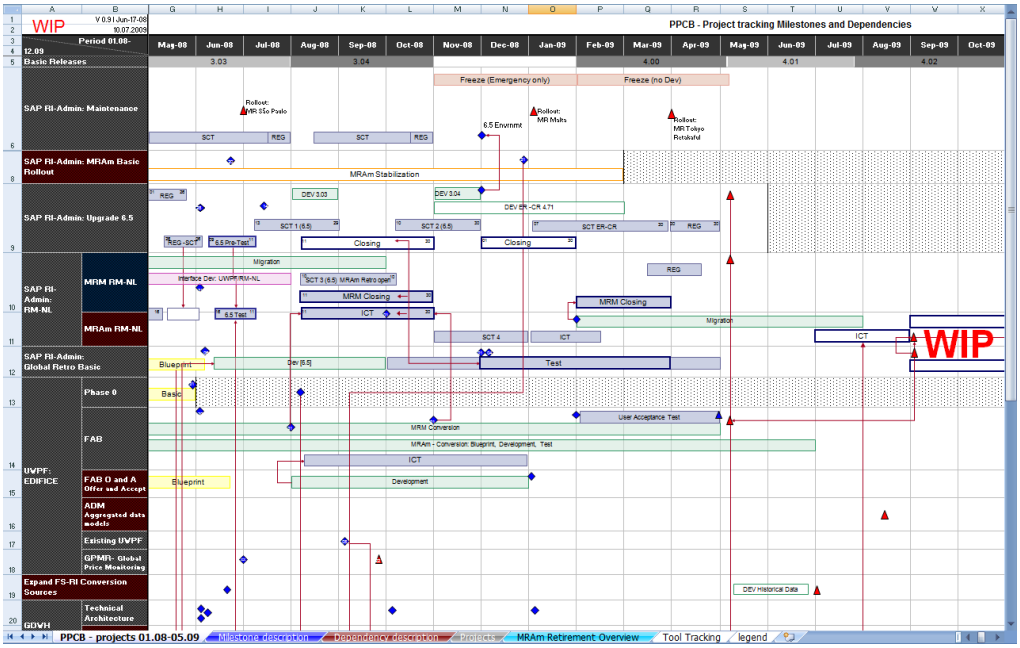


Figure 2.4: Snippet of the *Project Dependency View* of the current approach

Chikofsky and Cross defined **reverse engineering** in 1990 as:

> the process of analyzing a subject system to identify the system's components and their interrelationships and to create representations of the system in another form or at a higher level of abstraction [CC90].

Reverse engineering is strictly limited to a process of examination, and it is not a process of change or replication. If a system should be changed or re-built, the activity to perform is called re-engineering, which includes some form of reverse engineering followed by some form of forward engineering. The main subareas of reverse engineering are **re-documentation** and **design recovery**. According to Chikofsky and Cross, re-documentation is the *creation or revision of a semantically equivalent representation within the same relative abstraction level.* By contrast, in design recovery *domain-knowledge and external information* are added to the observations of the subject, in order to to identify meaningful higher level abstractions. The reverse engineering of Munich Re's current approach done in this section will cover both mentioned subareas. The modeling of the implicit underlying information model is a re-documentation task. But the creation of an entire legend explaining all included objects is a design recovery task, because it requires both, domain and solution knowledge. Therefore, the

main objectives are recovering lost information (legends) and generation of alternate views (information model), in order to cope with complexity. Additionally, reuse can be realized, for example, if the information needed for the roadmap is already stored somewhere else.

2.2.1 Stakeholder identification

A roadmap for enterprise architecture evolution might be used by different people from different divisions. The first step in the stakeholder-identifying-process is to determine possible users. This is done by an organizational structure analysis. Figures 2.5 and 2.6 show divisions and roles which might have used the current approach. The GBA department is the department mainly responsible for EA management. Nevertheless, they work intensively together with the IT strategy division. All identified groups of interest are listed and described below. Each description is derived from company-internal documents enriched by details from an IT architect.

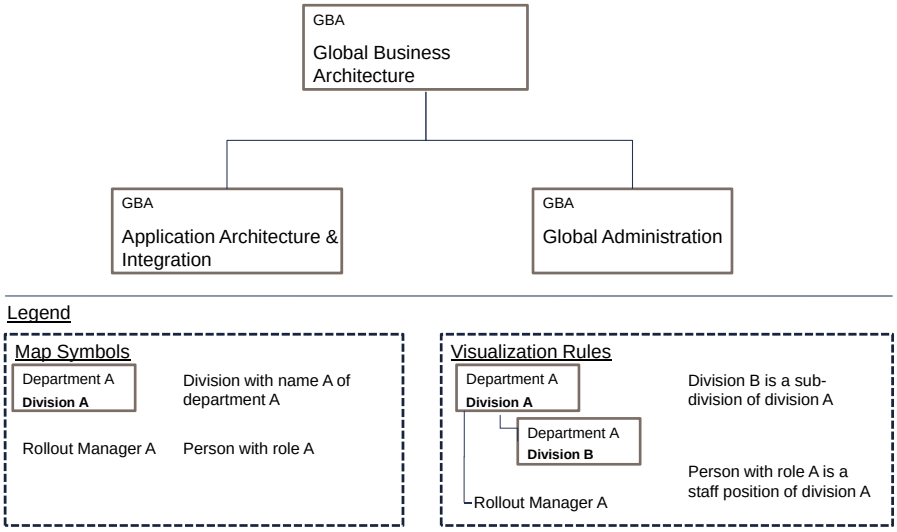


Figure 2.5: Organization chart of the GBA department

IT strategy The IT strategy division has several mandates. One of its responsibilities is sourcing. The people there evaluate, which IT tasks should be outsourced and which should be kept internally. Another responsibility is to support software project proposals with architectural designs. Since 2007, it also belongs to the tasks of the IT strategy division to implement an EA management approach and drive

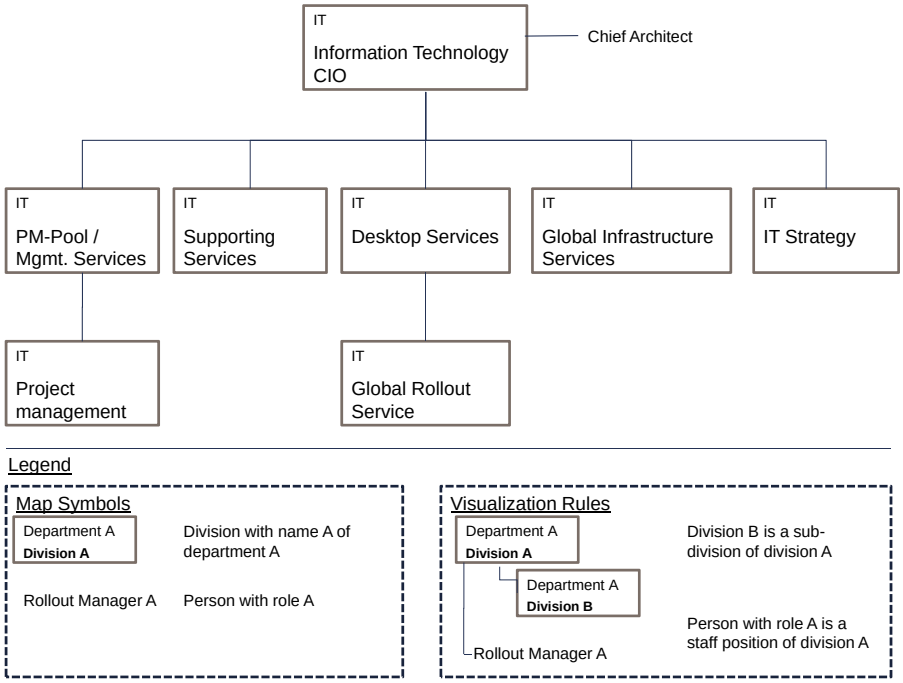


Figure 2.6: Organization chart of the IT department

its development. Other topics like compliance, sourcing, rollout scenario planning, and IT trends are also addressed by the IT strategy division.

Chief architect The chief architect is a staff position of the chief information officer (CIO). This person is responsible for the overall planning of the application architecture of the company and reports directly to the board of directors.

Global business architecture In 2008, a new division called global business architecture (GBA) was established to ensure a better alignment of business and IT. Therefore, the new unit constitutes the central interface between business and IT. GBA coordinates the prioritization of new project proposals including necessary deadlines and forms required as well as the assessment of IT budget. Additionally, it is responsible for an overall consistency of the individual business processes, as well as the business, reporting, and application architecture derived from them. Therefore, GBA drives the definition and implementation of the IT landscape based on the individual business processes. Because it also defines the target architecture, a roadmap for enterprise architecture evolution is essential.

Project management pool A project manager bears responsibility for

estimation, planning, steering, controlling, change requests, and risk management on the one hand and for team-leadership, team building, conflict management, communication, and collaboration for objective agreements on the other hand.

Global rollout services The global rollout service division is responsible for rollouts with client installation and release management. They plan resources for rollouts and do packaging and scripting for all rollouts world wide.

Global infrastructure services The vision of the topic leader team, within the global infrastructure services division, is to establish an architectural framework that will be the foundation for realizing a uniform IT platform for the Munich Re Group. The work from the topic leader team will enable IT to support the global strategic business goals as well as the local client-oriented needs of the MR Group. They define and initiate IT projects necessary to develop the technical architecture.

Global portfolio management The global portfolio management board consists of people from different divisions, for example IT and GBA. They try to establish a worldwide project portfolio management. Their focus is mainly on overall resource planning and delay detection.

The Excel document containing the current approach for release and rollout management was used by different people from different IT divisions. The chief architect used it to get an overview over current projects affecting the application landscape. For the same reason, the IT strategy division used the current roadmap. In addition, also project managers of projects visualized in the roadmap used it for overview reasons. Due to the lack of usability and reliability, people generally avoided the use of this roadmap. Therefore, the roadmap was not continually developed or maintained after it was first made.

2.2.2 Analysis of presentation

The current approach for release and rollout management consists of two main views, the *Project Dependency View* and the *Retirement Overview*. The other views within this document only show descriptions for objects

used in the main views. A collection of open issues is also integrated. In fact, the *Project Dependency View* and the *Retirement Overview* are instantiations of underlying viewpoints. Because there is no documentation available for these viewpoints, both views will be analyzed in this section, in order to create a documentation for visualization rules and included objects. The result can serve as a basis for the reverse engineering of the information model, which is described in the next section.

Project Dependency Viewpoint

The *Project Dependency View* shows an overview about IT projects for a two year planning period (see Figure 2.7). It shows project phases, milestones, and dependencies between different projects. The view is encompassed by some meta information like the current date and the date of the last change. A Cartesian map makes up the main map of this view. The horizontal axis is a timeline which is interval scaled with an interval size of one month. By contrast, the vertical axis is ordinal scaled and made up of labels representing development projects. To indicate how complex a rollout might be and which *Organizational Unit* will be involved, three different colors for the project label background are used to show if it is an US, a Munich or a global rollout. Projects can be enriched by a note which is displayed right next to the project's name.

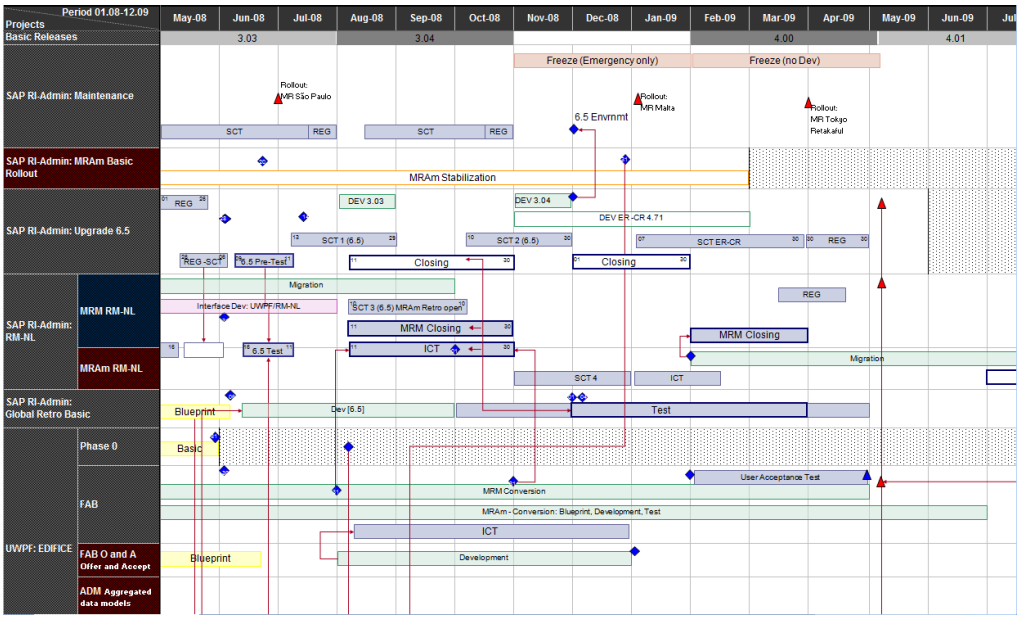


Figure 2.7: Snippet of the *Project Dependency View*

Within the map, different objects can be aligned to the horizontal and vertical axis. These objects represent milestones, rollout dates, phases, and dependencies and are located at the appropriate cells, depending on the project they relate to and the time they take place. Within the *Project Dependency View* exist three types of milestones, which are common project milestones, rollout milestones, and retirement relevant milestones. In addition to these milestones, each relevant phase of a project is shown as a colored bar, with its start day at its inner left and the end date at its inner right, in case they are defined. In order to make dependencies visible, red arrows are used to connect dependent milestones, rollout dates, and phases. They can also connect one object to several other objects and point in both directions.

Due to the fact, that there is no complete legend available and in order to summarize the previous mentioned insights, the legend shown in Figure 2.8 was reverse engineered.

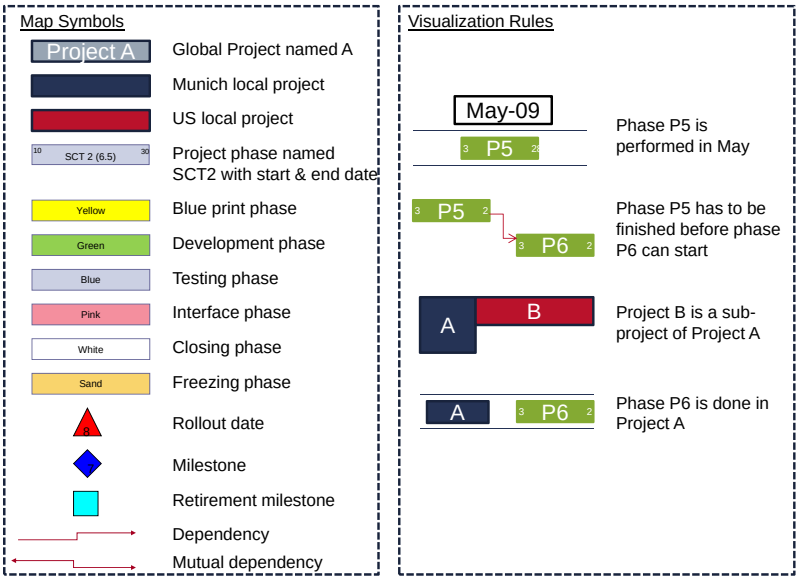


Figure 2.8: Reverse engineered legend for *Project Dependency View*

The legend shown in Figure 2.8, provides three different colors to highlight project influences to *Organizational Units*. *Global projects* are shown with a gray background, *Munich projects* with blue background, and *Princeton projects* with red background. In order to distinguish between different project phases, six different colors are used. A yellow background indicates that a phase is a *Blue Print phase* which means that conceptual work, like an architectural design, is done. A green background suggests that *Development* and implementation is done at this time. A blue project

phase background indicates that this phase is a *Testing phase*, for example, a user acceptance test or an integration test. A pink background marks an *Interface phase* where interfaces between applications are developed. A white background highlights a *Closing time*. Within this period, changes to applications are not possible, because they are needed to compile the financial statement. Finally, a sand colored background indicates *Freeze times*. They are comparable to *Closing times* but they relate to technical reasons.

Recapitulating we can hold down that the *Project Dependency View* is good as a first step but can be improved. The arrows are not aligned automatically, so they can cross each other and overlap project phases in an unpleasant way. Since also two sided arrows exist and some of them end in the middle of a project phase, the exact meaning of a dependency can be ambiguous in some cases. The only hint available is a short description for some of these dependencies which is stored in a separate view and not obvious for the user. The use of frame colors for project phases is not consistent. Normally, the frame has the same color as the background color, but sometimes it has a different color. Because there is no obvious reason for that, this fact can result in misunderstandings. Furthermore, the effort needed to create and maintain this roadmap by hand is not reasonable and leads to the need for a tool with a consistent information model and the ability to create a visualization.

Retirement Overview

In addition to the *Project Dependency View* a *Retirement Overview* is available in the previously mentioned Excel document. Figure 2.9 shows a snippet of it.

The matrix provided there has a simple structure. Its horizontal axis shows one column for each application, which will be retired. If it is a *Mainframe Application*, it is highlighted with a yellow background color. The vertical axis shows all applications which will replace one or more other applications. For each retirement, an `X` is set in the appropriate intersection of the corresponding application's column and row. If an application might replace another but it is not obvious yet, a `?` is used instead of the `X`. Because there is also no legend summarizing these information available, Figure 2.10 shows the reverse engineered legend for the *Retirement Overview*.

Application Retirement Matrix- April 1, 2008	ARS - Annual Reporting System	ADJ - Detailed Adustment System	OQR - Qrasp Quarterly Reporting	GRP - GRASP Reporting System	SFS - Schedule F System	APS - Accounts Payable Interface	ART - Accounts Receivable Treaty	BIS - Branch Interface System	CLP - Cooperative Loss Proc. (MainframeF rontend)	DWU - Deposit, Written and Unearned Estimates	FRP - Fac Rsk Processing	JIS - Journal Interface System	RCS - Retro Cash System	SICS - Storebrand Interantional Contract System	TIS - Treaty Information System	TPC - Treaty Premiums and Commissions	ATE - All Time Experience	CAT - Catastrophe Reporting System	CEX - Facultative Coded Experience	CLI - Client Information System	COM - Commission Pipeline	LDS - Loss Development System	LOS - Loss Pipeline System	PRM - Premium Pipeline System	TEX - Treaty Experience	CIP - Capacity Interantional Pool	CPS - Capacity Pool System	FAR - Facultative Accounts Receivable	GLS - General Ledger Interfaces	HPS - Hold Processing System	APSDMCA MGA AP Interface	BPS - Branch Planning System	GRIP	RAPS	VEWP (Data Warehouse - MFAm)	CRS	EMTEA
Projects Responsible for Retirement																																					
MRAm FSRI Basic Implementation							X	X		X		X	X	X	X	X		X		X		X									X						
Global Reporting/GDWH																																			X		
Global Reserving Platform/Process																																				X	X
Long Term Retro Solution					X												X		X						X	X	X										
RMNL	X	X	X	X	X	X			X		X						X				X		X	X				X	X		X	X	X				
UWPF FAB																																	X				
UWPF FAB - Future Phase																																					
EDIFICE - Aggregated Data Models																																					
EDIFICE - FAB O&A																																	X				
EDIFICE - Change Requests																																		X			

Figure 2.9: Snippet of the *Retirement Overview*

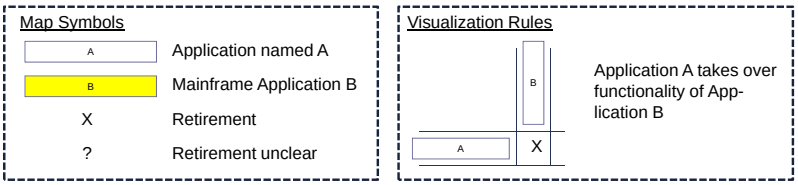


Figure 2.10: Reverse engineered legend for *Retirement Overview*

Due to the fact that one application can replace several other applications, and one application can also be replaced by more than one application, many `X` can occur for each column and each row. As a consequence, it is hard for the user to get an overview about all retirements. A visualization as a cluster map could be easier to understand for the user. Maybe some users also want to see all applications, which retire a specific legacy system, but also all applications which are retired by them, comparable to the transitive closure containing all retirements referring to a specific application directly or indirectly. Getting this information out of the *Retirement Overview* would be very time consuming.

Open Issues Viewpoint

As mentioned in the first paragraph of Section 2.2.2, the current approach also consists of a collection of open issues, called tool tracking, which can be seen in Figure 2.11. It is a simple list of questions and to dos, providing information about the person who raised the question, the date, the due date, the responsible people, and the current status. Each question or to

do in this list sets up a task and can also be enriched with a comment. For the further proceeding of the analysis, this part of the current approach will not be taken into account, because it is not directly a part of the roadmap and so out of the scope of this thesis. Nevertheless, it is a good idea to collect all upcoming issues related to the roadmap at one central location.

Type	Task	Raised by	Date	Due	Responsible	Status	Comments
Question	CSI - global or local project	L. Bell	Apr-08-08			Unknown	
ToDo	Global Match => Global Reserving	S. Berner	Apr-07-08	Apr-25-08	A. Joiko	Unknown	- consolidation of dependencies
Question	ICO BI project					Unknown	
Question	GPMR reports -> linkage with GDWH	T. Fiedler				Unknown	
Question	Interfaces - dependencies, conception (e.g. SAMBA)	B. Bindig	Apr-09-08	Apr-25-08	A. Joiko	Not planned	- clarification with Thomas F.
Question	Claims - Ultimate Estimation Management -> Pre-study and dependency to Global Reserving Platform; project (by GCNA 4.4) approved	J. Plenio	Apr-24-08	May-01-08	A. Joiko	Unknown	- clarification with PPCB
To Do	impacts on timing of UWPF to other projects e. g. US-Strategy	L. Bell	Apr-25-08	June-30-08	L. Bell, A. Joiko	On Time	

Figure 2.11: Collection of open issues

2.2.3 Analysis of contained information

The objective of this section is to reverse engineer an information model for Munich Re's current approach to EA evolution management. According to Müller et al. this task is considered as *data reverse engineering* [MJS+00]. In general, data reverse engineering tackles the question of what information is stored and how this information can be used in a different context. He also depicts that the analysis activity aims to recover an up-to-date logical data model that is structurally complete and semantically annotated. In most cases, important information about the data model is missing in the physical schema catalog extracted from the database. Due to the fact, that the current roadmap was made by hand without any database support, there is no explicit information model available which can be analyzed. For such a situation Müller recommends to ask developers, users, and domain experts because they can often contribute valuable knowledge. Therefore, this technique was used to reverse engineer the demanded information model.

The reverse engineered information model derived from the *Project Dependency View* is shown in Figure 2.12. The central class is the `Project` class representing IT projects. Projects can be grouped if they are sub-projects of a `ProjectGroup`. Each `Project` is linked to the `OrganizationalUnit` where it takes place. If no `OrganizationalUnit` is associated the respective `Project` is regarded to be global. For each `Project` all `Milestones` and `Phases` are determined and associated. Because they can depend on

each other, `Milestones` and `Phases` are summarized in their super-class `DependendObject`. To represent the hierarchy of project phases a composite pattern is used. Dependencies among `Milestones` and `Phases` are realized by the `Dependency` class. It always connects two `DependendObjects` and adds a description and the status. Information about responsible people is modeled by the class `Employee` which is associated with the respective `Project`, `Milestone`, or `Dependency` a person is responsible for. Notice that no responsible person is determined for project phases.

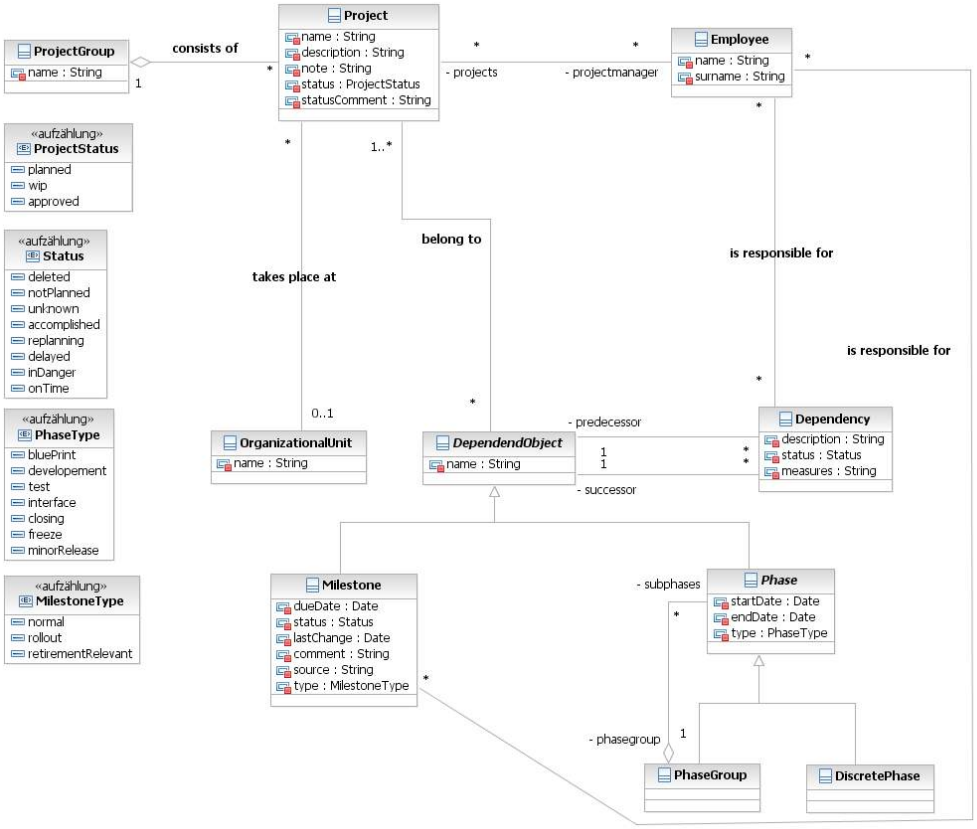


Figure 2.12: Reverse engineered information model for the *Project Dependency View*

The following list explains all classes and associations used in the reverse engineered information model:

Classes:

Dependency An instance of this class is used to represent a dependency between `DependendObjects`. It stores information like a description, the status (see enumeration-class `Status`), performed measures, and the type (see enumeration-class `DependencyType`).

DependencyType This enumeration-class lists all possible types of `Dependencies`. It distinguishes between a one-sided-dependency, which means, that the `DependendObject` in role `predecessor` has to be finished, before the `DependendObject` in role `successor` can start, and a two-sided-dependency, which additionally means, that the `DependendObject` in role *predecessor* can only continue, if the `DependendObject` in role *successor* has finished.

DependendObject This class is a generalization for all objects in the roadmap which can depend on each other, for instance one has to be completed before the other can start. Every sub-class, for instance `Milestone` and `PhaseComposite`, will inherit the attribute name which identifies each instance.

DiscretePhase Is a concrete and really performed phase during a `Project`.

Employee An instance of the class Employee always represents a physically existing person, who is employed by the company. Every employee has a name and a surname.

Milestone `Milestones` are important points in time during a `Project`. Therefore, they have an attribute dueDate, which represents the exact date of the milestone. Because a milestone can have a status, such as in danger or on time, the attribute status of type `Status` is added. The attribute lastChange is used to indicate the date when the last change of an attribute or dependency to a specific milestone happened. As usual, the attribute comment is used for comments. The attribute source links a `Milestone` to a document located somewhere outside the roadmap containing additional descriptions.

MilestoneType An enumeration of distinct types a `Milestone` can belong to.

OrganizationalUnit An organizational unit represents a subdivision of the organization according to its internal structure. A possible example are the entities showing up in an organigram.

PhaseGroup A `PhaseGroup` groups several `DiscretePhases`.

PhaseType This enumeration-class lists all possible types of `Phases`. They mainly cover the software life-cycle with types for development, testing, and freeze. A special type is closing, which is used for phases, which indicate the time when the old application is computing the annual accounts and therefore cannot be touched and no rollout can be done.

Project The class `Project` represents all planned or running IT projects.

ProjectGroup This class is needed to represent the hierarchy of projects. A `ProjectGroup` always consists of instances of the class `Project`, which are considered as sub-projects.

ProjectStatus This enumeration-class lists all possible states for `Projects`. A project can be planned, work in progress or approved.

Status This enumeration-class lists all possible states for `Milestones` and `Dependencies`. The status shows the satisfaction of the schedule. Only the status replanning indicates, that some aspects of the schedule will be redefined.

Associations:

Project takes place at OrganizationalUnit: Each `Project` is linked to the `OrganizationalUnit` where it takes place.

DependendObject belongs to Project: This association is used to indicate which `Milestones` and `Phases` belong to a certain `Project`.

In order to generate the *Retirement Overview*, another information model is necessary. The reverse engineered model shown in Figure 2.13 is sufficient to generate the *Retirement Overview*.

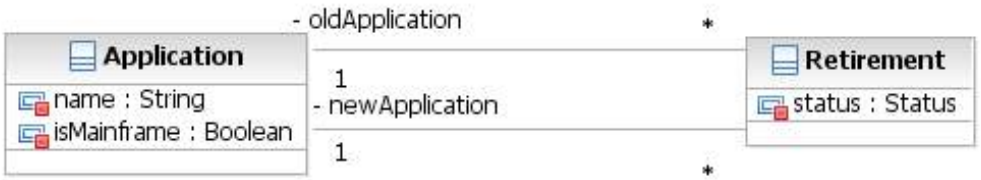


Figure 2.13: Reverse engineered information model for the *Retirement Overview*

Classes

Application The class `Application` represents all operating applications, also called legacy systems, and applications which are still under development. If an application is running on a main frame, the boolean flag isMainframe is set to true.

Retirement Every time an `Application` is going to fulfill at least one part of the functionality of another `Application`, an instance of the class `Retirement` is used to store this information. The attribute status is used to indicate the state of the retirement.

Associations

For the two associations between `Application` and `Retirement` no names are used. One association is used to associate the `Application`, which will be retired, in the role *oldApplication*. The other one is used to associate another `Application` in role *newApplication*, which will be operating in stead of the retired `Application`. Because one `Application` can retire more than one other `Application` and one `Application` can be retired by more than one `Application`, the multiplicity * is used for both associations.

Chapter 3

Requirements Elicitation

In the past 30 years, software requirements have been repeatedly recognized to be a real problem. Brooks stated in his classic paper on the essence and accidents of software engineering that the *hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements* [Bro86]. As Bell and Thayer observed in their empirical study, requirements are often incorrect, ambiguous, inconsistent, or simply missing. This takes effect in all kinds of projects [BT76]. The impact the requirements have on the resulting software is critical. Bell and Thayer concluded that requirements *do not arise naturally. Instead, they need to be engineered and have continuing review and revision.*

According to Bogner and Menz, expert interviews offer an unrivaled dense data collection and a straightforward access to the research area [BLM05]. For that reason, expert interviews have been conducted, in order to determine the requirements for a roadmap for enterprise architecture evolution. These interviews are designed as a mixture of questionnaire and open ended interviews. Questionnaire interviews appear scientific, but they are limited by their stimulus-response model of interaction, which assumes that a given question always has the same meaning to all subjects and excludes the kinds of interaction that could be used to establish shared meaning between the subject and the interviewer [GL93]. This disadvantage is weakened by additionally stated open ended questions, which tempt the subject to narration and allow less constrained interaction between the interviewer and the interviewee, in order to establish a shared meaning. That raises the question, if interview data from open ended questions is really applicable.

But according to Goguen, if *higher level structure is needed, then open ended interview data may be adequate* [GL93].

The focus for all interviews will be on determining functional requirements. Applying Brügge's categorization of requirements [BD04] into functional, non-functional and pseudo requirements, the non-functional and pseudo requirements are out of the scope for this requirements elicitation. The reason is that the resulting patterns should be only on a conceptual level and free of implementing details. Non-functional requirements, like usability, or pseudo requirements, like a specific programming language, will differ between companies and therefore prevent a general approach. The requirements verbalized in the following sections are already interpreted and categorized, in order to enhance comprehension and comparability. The approval of all interview partner's to the results shown in this chapter was given after a review. Towards the end of this chapter, the most important requirements are summarized and a gap analysis contrasting the identified requirements to the current approach can be found.

3.1 Interview design and setting

The requirements for a roadmap, visualizing the enterprise architecture evolution, can originate from several different divisions of a company. To identify these requirements seven interviews are conducted, with each interview partner from a different division. Among the interviewed people is an IT architect from the IT strategy division, the chief architect, a topic leader, an IT architect from the global portfolio management team, a project manager, a rollout manager, and a business architect from the global business architecture division. A detailed description of these roles can be found in Section 2.2.1. To ensure a widely spread perspective, the people asked for their requirements are from three hierarchy levels which can be seen in Figure 3.1. This figure shows only relevant divisions (as boxes) of the IT department and only roles of people which were relevant for the interviews (as text linked to a box). All interview partners are highlighted with a yellow background. If a division is highlighted, the head of that division was the interview partner.

In addition, Figure 3.2 shows the hierarchy of the GBA department. As Figure 3.1, it shows only that part of the organizational structure which was relevant for the interviews.

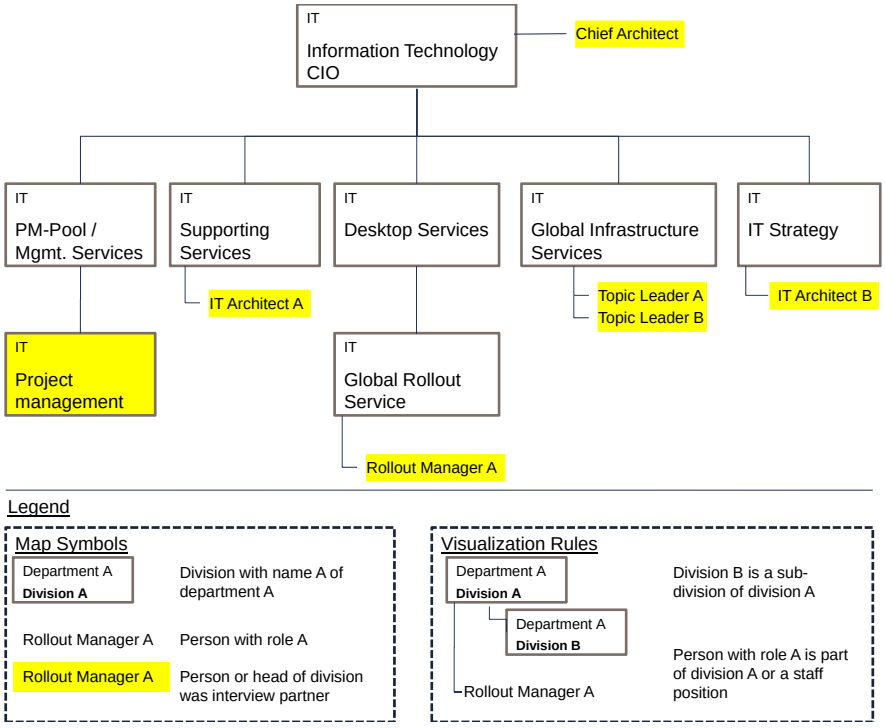


Figure 3.1: Organization chart of the IT department with highlighted interview partners

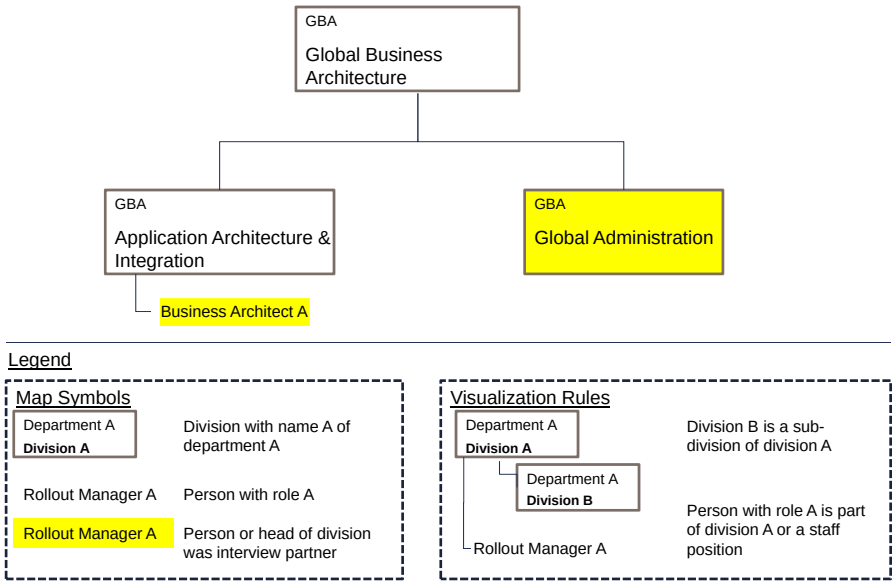


Figure 3.2: Organization chart of the GBA department with highlighted interview partners

The questions asked during the interviews are summarized in an interview guideline, which can be found in Appendix A. Beside questions about the interview partner's work and his or her experience, the guideline contains

questions to determine requirements about the needed information and about the preferred way of presentation. The guideline is subdivided into modules. Depending on the interview partner's work and experience, the appropriate modules are selected for the interview. If the interviewee already knows about the current approach for a roadmap, the module which contains questions about the current approach is included, otherwise it is excluded. The same applies to the module about the global portfolio management. Such an adjusted guideline was sent to each interview partner one week before the interview took place, in order to allow them to prepare themselves for the interview and so increase their responses quality. Figure 3.3 visualizes the different interview topics. Topics encompassed by brackets are optional and only included in interviews if the subject has the specific knowledge. The results of these interviews are explained in the following sections.

Figure 3.3: Interview topics and their sequence

3.2 Requirements from an IT Architect

In order to determine requirements from the IT strategy division a senior IT architect answered the questions of the interview guideline. To his area of

responsibility belong tasks like supporting consolidation projects with conceptual designs and developing the enterprise architecture. Furthermore, the IT strategy division is competent to support projects with software architects and is in charge for sourcing. The interview partner's interest is directed towards information about dependencies between different projects.

Interview Summary

For the IT strategy division it is important that every existing dependency is included in the roadmap. Dependencies have to be described explicitly in order to make them understandable even for people who are not involved in the corresponding projects. In addition, the dependencies have to be reliable, that means that every involved project committed to the dependency and is aware of the consequential impacts. The distinction between the date a project is able to deliver something according to a dependency and the date the other project needs the deliverable is important to compute lag times and bottlenecks. Due to the fact, that IT strategy is often interested only in projects on the main level and not on the sub level, they need a special view aggregating sub projects to main projects. In order to enhance readability, people from IT strategy do not need to see objects like milestones or phases which do not have any dependency. The distinction between major and minor releases is necessary because it leads to an estimation of the rollout extend. Minor releases are bug fixes or bring only little new functionality. By contrast, major releases always need a complete rollout process, which can include data migration or an application retirement. Because IT strategy is responsible for the strategic planning of the enterprise wide IT, the people need also planned projects and steering or consolidation initiatives visible in the roadmap. For the same reason, retirements of applications have to be described explicitly. This includes a description about the functionality and the data which will be adopted. Furthermore, also applications which will be retired have to find their way into the roadmap, because the exact dates of their retirement are essential for further planning. Moreover, a matrix representation of retirements is required. This matrix is not directly a part of the roadmap, but it is a prerequisite, because the planned retirements might be the reason for the initialization of new development projects. Additionally, a view showing the transitive closure of retirements for one or more applications is important to estimate impacts of a single retirement delay. The transitive closure

of retirements in this case would cover all applications which retire at least one part of a legacy system and in addition all other legacy systems which are also retired by one of these applications, and so forth. One rendering possibility would be a bipartite graph represented as a boxes-and-lines diagram. Figure 3.4 shows such a diagram as it was developed during the interview. Last, the planning process would be improved by the ability of scenario creation and impact simulation of additional projects.

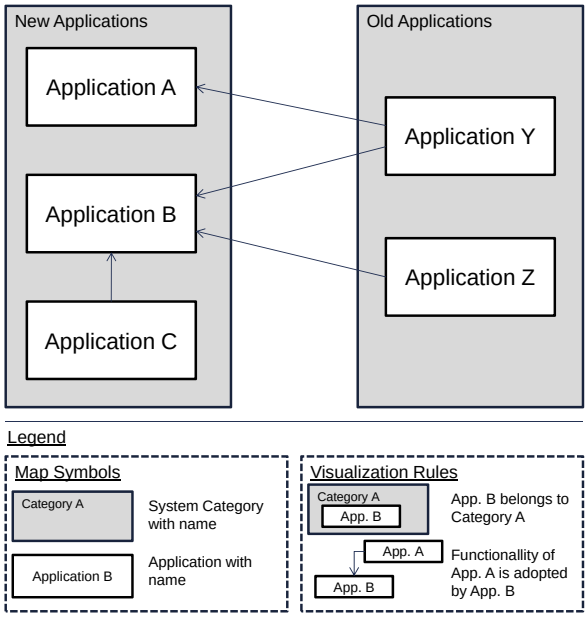


Figure 3.4: Diagram for retirement visualization

Derived Requirements

- For each dependency between two projects a dependency is shown in the roadmap
- Dependencies are described very explicitly and require a formal and committing agreement between the involved project managers
- Each dependency has two different dates; one for the time a project can deliver something and one for the time the other project needs the deliverable
- Filtering criteria:
  - phase hierarchy
  - objects with or without dependencies

  - initiatives and planned projects

- Duration of planning: 2 years
- Deliverables / milestones are categorized either as minor or as major release
- Retirements are described very explicitly and require a formal and committing agreement between the involved project managers
- Legacy systems are displayed
- The transitive closure of retirements is visible for one or more applications as a boxes-and-lines diagram
- Scenarios and simulations are available

Nice-to-have Requirements

In addition to the mentioned requirements, the senior IT architect mentioned, that the ability to save a snapshot of a current roadmap would be a nice feature. Monitoring of all changes made to the roadmap is not necessary, because the main function of the roadmap is planning for the future and not documenting the past.

Conclusion

Regarding the requirements raised by the IT strategy division, it becomes clear that these people are mainly interested in project dependencies. For them, seeing where dependencies occur is as important as the certainty that every dependency is exactly described and committed by both projects. Furthermore, the IT strategy division needs a roadmap only showing actually relevant phases and milestones but also planned projects, initiatives and retirements. Retirements, phases, and milestones always have to be planned with all dates in detail. In order to plan foresightful, the IT strategy division also needs scenarios and simulations.

3.3 Requirements from the Chief Architect

A roadmap showing the evolution of the enterprise architecture is the daily bread for a chief architect. He is responsible for the future appearance of the application landscape and of course, he needs a plan to get to the conceived landscape. His planning activities extend to the next six years.

Interview Summary

For the chief architect, a high level overview of projects, milestones, and dependencies will be appropriate in general. Sometimes he also wants to see some project in detail, so the roadmap has to have a filtering function for sub-projects. Because he is believed to have the general view over all projects, he needs a short description of each project shown in the roadmap. Furthermore, interfaces and necessary hardware has to be described in detail by each project. In order to make changes to the roadmap easily comprehensible and maybe even able to revoke, the roadmap has to be under version control. Additionally, dependencies have to be categorized, either they depend on data from another project or they use functions from another project. This is important, in order to understand the dependencies. For example, functions can be provided at the time their development has been finished, but data has to be loaded or even enhanced, which can take several days. Because the chief architect reports directly to the management, he also needs a graphical representation of a relevant part of the application landscape taking actual project into account. This top-level overview should only show main applications grouped as platforms and aggregate dependencies between them. This overview can be used to develop the vision of the application landscape and give the management an overview about what is happening.

Derived Requirements

- Filtering criteria: project hierarchy
- Duration of planning: 6 years
- Projects and tools have a short description
- Applications have descriptions of their:

  - interfaces
  - hardware they use

- Changes to the roadmap are comprehensible and therefore under configuration management
- Dependencies are categorized
- Visualizations at top level, showing only components/platforms and where general dependencies exist on that level, is available

Conclusion

In summary, the chief architect is mainly interested in a general overview, but sometimes also in information about technical issues. Short descriptions of objects used in the roadmap are essential. The view must be configurable to meet his needs, which are on the one hand his daily work and on the other hand his reporting activity to the higher management.

3.4 Requirements from a Topic Leader

Topic Leaders are responsible for Technical Architecture which is part of the Enterprise Architecture for the Munich Re Group, for example, global network, physical / virtual servers and basic security issues. They have their own roadmaps showing which baseline application, for example, operating systems and databases indicating specific version will be in service for the future. Therefore, they have to adjust their planning according to the different vendor's software and hardware, their type of support and to the application architecture which can require specific infrastructure.

Interview Summary

For this interview two topic leaders were concurrently asked what their responsibilities are. Both mentioned, being aware of every maintenance weekend during rollout planning for applications is very important, because maintenance activities can make a rollout shift necessary. Therefore, the rollout dates of infrastructure services, such as operating systems,

databases, and office tools have to be included in the roadmap for the next two or three years.

Derived Requirements

- Rollout dates of software considered as infrastructure is part of the rollout planning within the roadmap
- Duration of planning: 2-3 years

Nice-to-have Requirements

It would also be extremely useful if the project managers would map their applications to the infrastructure they need. This has to be done during the projects planning phase, in order to ensure the timely availability of the appropriate infrastructure services.

Conclusion

To sum up, topic leaders have a strong demand on them to publish accurate roadmaps to support the integration between the planning of infrastructure and application rollouts.

3.5 Requirements from a Global Portfolio Management IT Architect

The people involved in the Global Portfolio Management (GPM), which is a new initiative operating since March 2009, try to establish a global project portfolio management. Among the participants are people from the IT, GBA, and Global Process Owners (GPO). GPOs are responsible for business processes. They define the processes, develop them and rise requirements for supporting applications. As a representative, a senior IT architect, who is also responsible for the architecture of all baseline SAP applications and the services they offer to other applications, summarized their requirements as follows.

Interview Summary

The interview shows immediately, that the mentioned requirements are from a managerial point of view. They are less detailed but give an overview of the purpose of the roadmap as part of a multi-project management. If project phases and inter-project dependencies are modeled, the critical path over all projects can be computed and visualized. Then, the impact of deviations can be determined more simply. Another benefit of the roadmap should be the ability to manage global resources across different projects. This is critical because delays at resources also requested by other projects can have crucial impact to these other projects.

Derived Requirements

- Dependencies between projects are identified and collected
- Critical paths are identified
- Deviations are identified early to find problems and solutions
- Overall key resource management is prepared and tracked
- The roadmap does not dictate individual project plans
- Simulation functionality is included
- Duration of planning: 2 years

Conclusion

For the GPM team, the roadmap is considered only as a support for each individual project manager and not as an overall planning instrument which forces projects to adjust their plans. The roadmap should be used as a basis for resource management and to identify critical paths.

3.6 Requirements from a Rollout Manager

The global release management is done within the global rollout service division, which is responsible for the coordination of all global rollouts.

For the rollout manager, the focus is on software, which needs a client installation, because server-side software releases do not need to be planned for each organizational unit across the globe individually. In addition, rollout managers are also a contact point for project managers respecting questions about rollout times, proceedings or packaging.

Interview Summary

In the eyes of the global rollout service division, a roadmap for enterprise architecture evolution has to include detailed information about closing and freeze times. Because these times differ from organizational unit to organizational unit, they have to be shown for each unit separately. To identify dependencies between organizational unit specific closing times and application rollouts, projects have to determine exactly where they want to roll out. Furthermore, each project has to determine the way it wants to deliver its application to clients, for example by the use of delivery tools or by manual installation from discs. The way of delivery can have crucial impact on the rollout duration and time frame. If a rollout has any special attributes, they have to be visible in the roadmap and communicated to the clients. For rollout management it is also important, to distinguish between the date a project is ready to roll out, the date the rollout has to be completed, the date an application is planned to operate and the date it has to be in operation, because other applications depend on their functionality. But the scope goes even further. To plan following versions and their releases, the expected date of retirement is also important. Due to the fact, that a rollout manager mainly has to handle information about the next three months, a filter should provide the ability to view only that part of the roadmap. Furthermore, the rollout management also requires detailed project plans, especially for testing issues, so a link to these documents should be provided by the roadmap. In order to avoid double work and conflicting inputs, the responsibility of data maintenance has to be obvious for every user of the roadmap.

Derived Requirements

- Closing and freeze times are included for each organizational unit
- Projects determine exactly where they will roll out their application
- The technique of delivery is described

- Rollout special attributes are described
- Planned rollout date is visible
- Planned and required operating dates are determined
- Distinction between ready for rollout dates and required rollout dates
- Expected retirement of applications is visible
- Filtering criteria:
  - timespan
  - organizational unit
- Links to documents detailing roadmap objects are included
- Explicit responsibility of roadmap data maintenance
- Duration of planning: 2 years, focus on the next 3 months

Nice-to-have Requirements

In addition to these requirements, the ability to develop compensational scenarios would be a nice feature. These scenarios could be used in case of rollout failures, in order to reduce reaction time. Moreover, it would be helpful, to visualize projects which are ready for rollout, and which projects are not.

Conclusion

In summary, it can be stated that the global rollout service is mainly interested in planned and required rollout dates and the involved organizational units. The distinction between the date a project is ready for roll out, the date it will roll out, and the date the roll out must be completed is very important for them.

3.7 Requirements from a Business Architect

The global business architecture (GBA) division is responsible for the evolution of the enterprise architecture from a business point of view. The following requirements were raised by a business architect and a head of division from the GBA during an interview. Because of some overlapping in the given answers during the interviews, which were done in sequence, both interviews are summarized in this section.

Interview Summary

The people from the GBA division need a roadmap for enterprise architecture evolution also in a pre-state of projects. Because they are responsible for the approval of projects, they need a roadmap showing issues which will be addressed in the next five years. An issue, in their definition, is a relatively widespread term. It can be compared to a change request for the enterprise architecture. For example, the consolidation of two or more applications actually providing the same functionality or the replacement of local applications by global applications would be issues. With these issues as a basis, projects with a better integration to the enterprise architecture can be initialized. This might be the only way which prevents, that each issue comes to an individual project. Therefore, people from GBA need a roadmap which contains issues and their dependencies. If there is a time frame already defined, important milestones should also be visible. Because the handling of projects is done in other divisions, a contact person has to be defined for each issue or project in the roadmap. The critical path should also be provided on the level of issues. Each issue has to document, which business processes will be touched. If there are any new interfaces expected, they have to be described in detail. In addition, each issue or project description should be enriched by data about technical infrastructure. For example, an issue should state, that for its specific data storage functions a database is appropriate and not a spreadsheet. Due to the fact, that application planning is derived from business processes, issues and projects should also be linked to the process they support. About the retirement of legacy systems, the roadmap should provide data about the retirement date and the business functions of the legacy system. To be aware of dependencies resulting from retirement processes, the transitive closure should be visualized for one or more applications. Last, the roadmap has to be un-

der version control, in order to make changes comprehensible and to build binding versions.

Derived Requirements

- Issues, projects and their dependencies are shown
- Each issue/project needs a responsible person who acts as a contact person
- Duration of planning: 5 years
- Critical paths are identified
- Support of specific business processes is visible
- Different scenarios are supported
- Assumed interfaces are described
- Important milestones are shown
- Issues/projects are prioritized
- Technology used by issues is determined
- Filtering criteria:
  - major or minor release
  - domain
- Retirement relevant information is provided (dates, business functions)
- The transitive closure of retirements is visible for one or more applications as a boxes-and-lines diagram
- Roadmap has to be under version control

Conclusion

Regarding the requirements raised by the GBA division, it is fair to say, that people there are mainly interested in a roadmap which shows issues and not already existing projects. If a specific project is already running, their work is usually done. Nevertheless, a roadmap on the upstream issue-level instead of the project level is very important, because it is needed to define projects in a way, that they fit into the EA. In addition, projects would then be supported by the determination of interfaces and dependencies to other projects in a very early phase.

3.8 Requirements from a Project Manager

In order to determine requirements from the project management the head of division answered the questions from the interview guideline. He is a former project manager and now responsible for the advancement of project methodologies, project initiation and execution. Project managers in general are responsible planning, handling, staffing and risk management during a project or a sub-project. Because the roadmap is concerned to show planned and running projects, the requirements from the project management listed below are essential.

Interview Summary

As expected, for project managers (PMs) the main purpose of a roadmap for EA evolution is a summary of currently running and planned projects and their dependencies. Nevertheless, pre-studies and initiatives should also be included, because they can have impact to other projects. It is also important, that dependencies and interfaces to other applications are not determined during a project. They have to be identified before the project kick-off and then documented in the roadmap. Therefore, the single scope of each project has to be determined in detail before any time-related plan is made or a rollout date is defined. To support the planning of the technical infrastructure and to ensure its availability, the required infrastructure has to be determined by each project. The roadmap also has to show the impact of delays of critical milestones, in order to enhance understanding of the global project portfolio among project managers. From the project

management point of view, it is important to prioritize projects and visualize this in the roadmap. PMs usually need a high level of detail for their planning. Therefore, the roadmap should visualize projects and their phases as a hierarchy and also support filtering by project priority, because often only the highly prioritized projects are important for future planning. In addition, the release management has to be included in the roadmap, especially all dependencies to SAP products, because they have a fixed release calendar and so other applications have to align to it. Furthermore, information about the retirement of legacy systems is important for project managers. It is not only the information that a retirement has to be done by a specific project, but rather the awareness of impacts caused by a delay in retirement. To allow a short reaction time in worst-case scenarios, different roadmap scenarios should be built. If major changes occur to the roadmap, a new version should be created and communicated to all stakeholders.

Derived Requirements

- Planned and running projects are shown
- Pre-studies are included
- Initiatives are included
- Roadmap is management approved
- Inter-project dependencies and interfaces are determined beforehand
- Project scope is determined and approved
- Needs of technical infrastructure are documented
- Impacts of delays to other projects are visualized
- Projects are prioritized
- Filtering criteria:
  - main and sub-projects
  - phases and sub-phases
  - project priority
- Duration of planning: 2 years
- Release dates are shown

- Retirements are shown with date and functions
- Different scenarios are supported
- Roadmap is under version control

Conclusion

Because project managers are mainly responsible for the handling of projects, they do not need a roadmap with strategic character for their daily work. For them, the benefit of a roadmap would be the ability to do resource estimation on a higher level and to derive a project portfolio management from it. Because the roadmap frames all projects, the reliability of its data and its feasibility is very important for project managers.

3.9 Summary

As a general result of the requirements elicitation, this section summarizes all requirements mentioned during the previous described interviews. In addition, the requirements are categorized in order to prepare the gap analysis in the following section. The categories used to summarize the requirements are *objects to be included*, *filtering criteria*, *visualization and calculation*, *managerial requirements*, *configuration management requirements*, and *simulation scenario requirements*.

Objects to be included

R1: **Issues** with link to business process, assumed interfaces, priority, and used technology

R2: **Projects** (planned and running) with needs of technical infrastructure, all organizational units where they are going to roll out, the technique of delivery, special rollout attributes, the ready to rollout date and the required rollout date, the planned and required operating dates.

R3: **Milestones** with a due date

R4: **Phases** with a date for the beginning and a date for the end

R5: **Dependencies** categorized by type, and with one date for the ability to deliver and one date for the obligation to deliver

R6: **Applications for retirement** also known as legacy systems, with a date for their retirement

R7: **Initiatives** as long as they have impact on other projects

R8: **Pre-studies**

R9: **Delays**

R10: **Closing times** for each organizational unit individually

In addition to the specific mentioned information about each object, all objects have to have a description and a responsible person. There also has to be the possibility to add links to documents containing detailed information to each object in the roadmap.

Filtering criteria

R11: **Project hierarchy** expand or collapse main projects to see or hide sub-projects

R12: **Project priority** hide all projects which are below a specific priority

R13: **Phases and sub-phases** expand or collapse main phases to see or hide sub-phases

R14: **Major or minor release** see or hide all projects which lead just to a minor release

R15: **Domain** hide all projects which do not affect a specified domain

R16: **Timespan** hide everything which will take place outside the specified timespan

Planning period duration

R17: 3 months

R18: 2 years

R19: 5 years

R20: 6 years

Visualization and calculation requirements

R21: **Critical path** has to be computed

R22: **Retirements** have to be visualized as boxes-and-lines diagram

Managerial requirements

R23: Inter-project dependencies require a formal and committing agreement between involved projects

R24: Retirements require a formal and committing agreement between involved projects

R25: Roadmap is for support only and does not dictate individual project planning directly

R26: Explicit responsibility of roadmap maintenance

R27: Roadmap has to be mandatory and therefore approved by higher management

Configuration management requirements

R28: Version Control

R29: Snapshots

R30: Change management

Simulation scenario requirements

R31: **Scenarios** show different possibilities for the roadmap

R32: **Simulations** visualize the impact of a project delay or a planned project

3.10 Gap Analysis

The purpose of this gap analysis is to identify conflicting requirements in the first step. This needs to be done, because conflicting requirements demand further attention. Normally, one needs to decide, which one of the conflicting requirements is more important. An implementation will focus on the satisfaction of that requirement, in stead of the other. The second step of this gap analysis is to determine gaps between the new requirements and the current approach. These gaps need to be bridged by a new approach, which could be derived from the patterns documented in Chapter 4.

3.10.1 Contrary Requirements

In general, two requirements are considered as conflicting, if the satisfaction of one requirement directly results in negligence of the other one. Examples are the requirements *high security* and *high performance*. A software system usually can not fulfill both requirements at the same time, by unchanged costs and resources. In order to enhance security, more data has to be processed due to authentication and authorization. This impacts directly the performance of the system in a negative way, because more resources are consumed. The following paragraphs describe similar conflicts between the requirements for a roadmap for enterprise architecture evolution.

One major conflict exists between the requirement to show issues (R1) and the requirement to show projects (R2). Of course, one might consider a roadmap showing both of them, but the complexity of the roadmap would increase intensely. In addition, the two objects issue and project originate from different levels of abstraction and are located next to each other in the process of roadmap creation. Usually, projects will be instantiated according to a previously identified issue. If the complexity of the roadmap should be kept as low as possible, the only way to address this contrast is to have two roadmaps on different levels, one showing issues, and one showing concrete projects.

Another conflict can be found regarding the requirements for the duration of the planning period of the roadmap. The shortest duration required is a period of 3 month (R17), the longest a period of 6 years (R20), which is more than twentyfold longer. But this conflict can be solved by a filtering

criteria for the shown timeline, as long as every user knows, that dates determined for over three years in future might not be fix and likely change. An appropriate duration for the planning period are 2 years, because it is in the middle of the required durations and it is also proven practice, because the current approach also covers a 2 years period.

The last conflict, which has been identified, is the appropriate level of configuration management. Some divisions only require the ability to take a snapshot of the roadmap, while others request a version management or even a change management. The contrast in this case is between the requirement of high level configuration management and the requirement to keep additional costs low, because every additional management activity occasions cost. At this point in time, this contrast cannot be addressed. The management has to decide which level of configuration management is appropriate for the roadmap. Without determining the level of configuration management for a general approach, it allows more flexibility. As a suggestion, at least a version management might be sensible, while change management should be done roll-based and not by an approval process.

As a result, the requirements for a roadmap for enterprise architecture evolution are complex. They are widespread and some are even contrary. In order to develop a general approach, the most important requirements need to be addressed and contradictions need to be solved. Therefore, the scope has to be scaled-down by reducing the number of addressed requirements. This reduction is done in Section 3.11. Every requirement, which will not be addressed anymore, is itemized there completed by a rationale.

3.10.2 Gaps between current approach and new requirements

The requirements elicitation brought up several requirements, which are not satisfied by the current approach. These gaps between the current approach and the new requirements are identified in this section.

The first major gap exists between the objects that are displayed in the roadmap. The current approach shows only projects and initiatives. By contrast, requirements R1, R6, R8, and R9 also request for issues, legacy systems, pre-studies, and delays. This gap needs to be bridged by a new approach, taking these additional objects into account.

The second major gap exists between the required details for each roadmap object and the currently shown details. Beyond the included details, projects for instance, should also define their technique of delivery, all organizational units where they want to roll out, the technical infrastructure they need and the date, when the application has to be operating. Also closing times should be displayed in more detail, for example, they should be defined for every organizational unit.

The third major gap is between the required filter functions and the filter functions available in the current approach. Requirements R11-R16 request for different filters, in order to hide roadmap objects which are not of interest for the specific user. The current approach does not satisfy any of these requirements. It offers no possibility to hide or show specific roadmap objects. The main reason why filters are essential for a roadmap is to reduce complexity. With filter options, the user can select what he wants to see and is not overwhelmed with unnecessary information.

In addition, some other gaps exist between the mentioned requirements and the current approach. The required possibility to link documents to roadmap objects is addressed partially. In the current approach, only milestones can be linked to external documents. Furthermore, the requested critical path (R21) is actually not visible. The possibility to create different scenarios (R31) and simulations (R32) is also not available. Really critical is the fact, that there is no configuration management (R28-R30) established for the current approach and no responsible person is assigned. Equally, the managerial requirements (R23-R27) are not satisfied by the current approach. This might be the reason why they were mentioned during the interviews, although they can not be satisfied by the roadmap, because they request for a process for managing the roadmap.

In summary, many gaps exist between the current approach and the mentioned requirements. Some of them will be eliminated by the limitation of requirements (Chapter 3.11), where the most important requirements are determined. Others need to be bridged by the patterns documented in Chapter 4.

3.11 Limitation of Requirements

Taking all requirements mentioned in the previous summary of requirements into account, one can say, that the requirements are relatively

widespread. They vary widely in the set of objects to be visualized and the different levels of abstraction. Therefore, it would be hard to satisfy every requirement at the first step, because some of them are even contrary, as shown in Section 3.10.1. Because the pattern-based approach to EA management provides one EAM Pattern for one problem and lets the user combine them to an integrated approach, it is the ideal way to begin with one part of the mentioned requirements and extend it later by new patterns. Because of that, a sensible range of requirements needs to be selected for the upcoming pattern documentation. This section shows, which requirements will not be addressed by the patterns documented in Chapter 4 followed by an explanation why these requirements are excluded.

The first requirement which is excluded is the requirement to show issues (R1) in the roadmap. By this exclusion, the gap identified in Section 3.10 is eliminated. The focus of this thesis is a roadmap showing actual projects, because projects are formal and approved. By contrast, issues are less formal and usually still under development, which causes frequent changes to the roadmap. Without any formality, issues cannot be stored in a database and cannot be assessed to display them in any visualization.

Second, the requirements R23-R27 covered by the category *managerial issues* are excluded. These requirements do not describe required functions of a roadmap for enterprise architecture evolution. Instead, they address the handling of the roadmap (R25) and its approval (R27). In addition, R26 requests for a role concept for roadmap maintenance. R23 and R24 request a process for the roadmap and not one of its functions. The only way to satisfy them is by management order or a workflow support. In order to develop a general approach, both can not be determined in detail. Therefore, only suggestions can be given.

Third, all requirements within category *configuration management* will not be addressed. As the gap analysis brought up, the appropriate level of configuration management has to be decided for each company individually. Because there is no direct dependency between the configuration management and the other functionality of the roadmap, it can be included as additional function later.

Last, the requirements R31 and R32, requesting the ability to develop different scenarios and do simulations, are also excluded. The reason is that both are complex functions and therefore they have to be described in more detail, which was not done during the interviews. Nevertheless, this would be a good point to start the further development of the roadmap.

3.12 Discussion of Interview Proceeding

To complete this chapter about the requirements elicitation done by interviews, the proceeding of the interviews will be discussed. The first problem was to identify the appropriate people to ask. As a basis, the organizational structure in addition with some responsibility descriptions allowed a first selection of interview partners. Moreover, the identified stakeholders of the current approach described in Chapter 2.2.1 were taken into account. Supported by an IT architect from the IT strategy division, it was possible to identify the right people to ask. The next problem arising at this point might be the ability to meet the identified people. Their offices are spread over different locations within Munich. Due to the fact, that some interview partners are from the management level, it might be challenging to get an appointment for one hour for an interview, because these people are usually very busy. In fact, this problem did not occur. All people invited to an interview, independent from their hierarchy level, were willing to answer the questions in order to determine requirements for a roadmap for enterprise architecture evolution. Perhaps the topic is critical in their opinion. One problem was that some people did not say everything on the point. But this problem was solved by the interview guideline's structure, which allowed people to narrate, but also included specific questions in order to extend answers by topics not covered by the narration. For the same reason, a certain amount of understanding of the specific domain was required by the interviewer. This problem was solved by the prior analysis of the current approach and the related work. Because the interview partners in general used the same expression for the same things, there was no risk of misunderstanding and the answers were comparable to each other. In summary, the proceeding of the interviews implied some challenges, but due to the preparation done before, no unexpected problems occurred.

Chapter 4

Applying the Pattern-Based Approach to EA Management

This Chapter presents all EAM Patterns for enterprise architecture evolution documented during this thesis. Therefore, first the three relevant types of EAM Patterns are presented. Second, a pattern map is provided to visualize the relations between the patterns documented in this thesis and links to EAM Patterns documented in the *EAM Pattern Catalog*. Afterwards, each pattern is described in its own section.

4.1 Pattern Structure

> An EAM Pattern is a proven practice-based, general, reusable solution to a common problem in EA management, for a given context, identifying driving forces, denoting known usages, and consequences [Ern07].

In detail, there are currently four types of EAM Patterns, namely *Methodology Pattern* (M-Pattern), *Viewpoint Pattern* (V-Pattern), *Information Model Pattern* (I-Pattern), and *Anti Pattern*. In this thesis only three types of EAM Patterns are used. Anti Patterns were not documented in this thesis because the target is to reverse engineer and document a proven practice approach and not approaches which did not achieve their objective. The following paragraphs provide a definition of the used EAM Pattern types according to [Ern07].

M-Pattern

Methodology Pattern (M-Pattern): A methodology pattern documents a proven practice solution to a recurring problem for a specific context in form of a process for the management of an enterprise architecture. It also documents roles, the steps to be taken in the process, inputs and outputs of the process, as well as known variants, and consequences related to its usage. The documented process can use one or more viewpoint and information model patterns during its execution.

V-Pattern

Viewpoint Pattern (V-Pattern): A viewpoint pattern documents a proven practice solution to a recurring problem for a specific context in form of a viewpoint for the creation of views. It also documents techniques for view creation and usage, as well as known variants, and consequences related to its usage. The documented viewpoint is a representation or input method for information, which can be stored according to one or more information model patterns.

I-Pattern

Information Model Pattern (I-Pattern): An information model pattern documents a proven practice solution to a recurring problem for a specific context in form of an information model fragment for the creation of an information model. It also includes definitions and descriptions of the used information objects, documents techniques for information model fragment implementation and usage, as well as known variants, and consequences related to its usage.

General Structure

Each EAM Pattern has a fixed structure to foster the comparability and selection of patterns. Table 4.1 summarizes all sections of an EAM Pattern and describes their purposes. In case where there is no sensible information to add in a specific pattern section, this section will be left out in the respective pattern.

Section name	Description
Example	An example illustrating the problem to be addressed by the pattern. This example should be used by the other parts of the pattern.
Context	The situations in which the pattern may apply.
Problem	The problem a pattern addresses, including a discussion about its associated forces. Only one problem per pattern. Forces are goals and constraints, which occur in the context.
Solution	The fundamental solution principle underlying the pattern.
Implementation	Guidelines for implementing the pattern. These are only suggestions. E.g. the need to introduce a special board in an organization, a person who has to take care about the creation and the timeliness of a view, or the need to implement a process for data collection, etc.
Variants	A brief description of variants or specializations of a pattern.
Known Uses	Examples where the pattern was used, e.g. usage in companies, tools, books, etc. Due to the restricted maturity of the field of EA management, some patterns are known under different names in different companies. A list of these synonyms can be given in this section.
Consequences	The benefits the pattern provides and any potential liabilities.
See Also	References to other patterns solving similar problems, and to patterns that help to refine the pattern under consideration.
Credits	Credits to other authors, reviewers and shepherds of the pattern. This section is important because only a community process guarantees that patterns constitute proven solutions.

Table 4.1: EAM Pattern form

4.2 Pattern Map

In order to help users in the selection process of EAM Patterns, each pattern lists all related patterns concerned with the related problems or providing alternatives. A pattern map visualizes these relations to provide a coherent overview. Using the pattern map appropriate EAM Patterns can be found easier because every pattern is linked to respective concerns addressed by the pattern, which can be used as an entry point for the pattern selection

process. In Figure 1.1, a snipped of the pattern map showing EAM Patterns included in the *EAM Pattern Catalog* was already shown.

Figure 4.1 shows a pattern map for the patterns documented in this thesis. They are located in the center of the map and linked to each other via solid lines. Around the new patterns, existing EAM Patterns from the *EAM Pattern Catalog* are linked to a new pattern if they address a similar concern or use similar visualization rules.

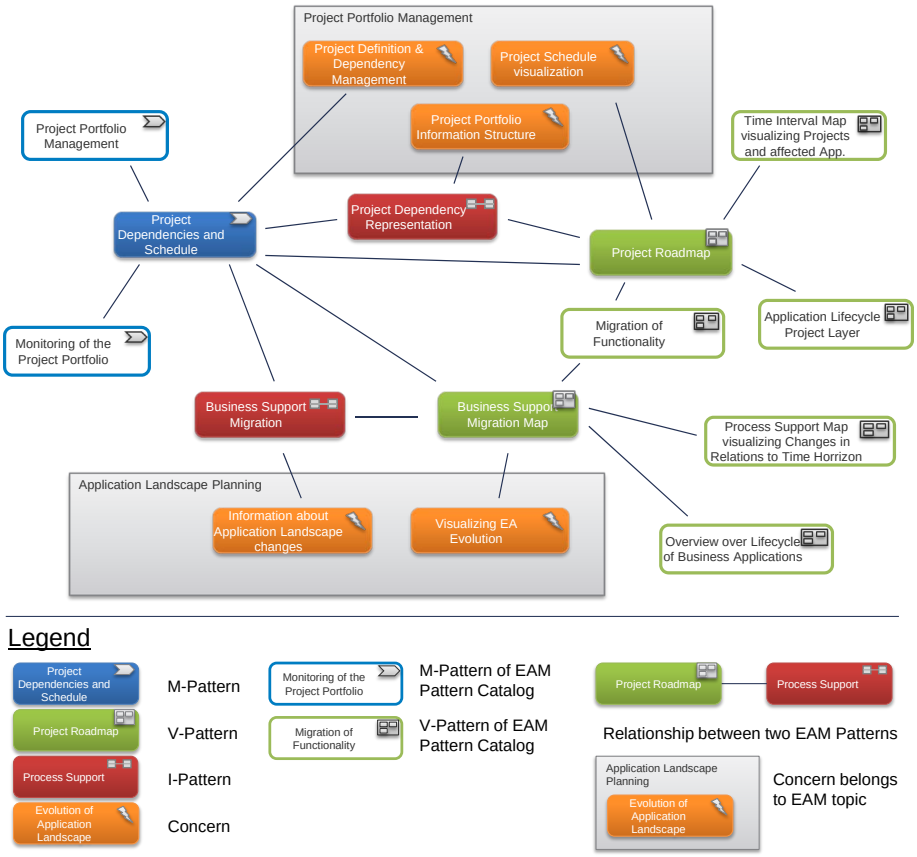


Figure 4.1: Pattern Map of newly documented EAM Patterns and their relationships to existing EAM Patterns

4.3 Project Dependencies and Schedule

The objective of Project Dependencies and Schedule is to define projects effectively and identify all important dependencies between current and planed projects, in order to schedule projects and react on delays timely. As a result, risks can be discovered more easily and projects do not exceed their budget.

Example

The MRAm project was considered to roll out three newly developed or adapted applications at a specific organizational unit. The reason was, that all organizational units around the world should use the same global applications and not their custom built systems. The first application was called FAB and was responsible for contract initiations with customers. The second was called RM-NL and was responsible for contract administration. The third was called FS-RI and was responsible for the generation of the annual accounts. In this order data included in contracts had to pass these systems. Because of the high complexity of the supported business processes, each application was developed in a separate project. Of course, project managers were aware that their application has to import data from another application. But initially, they did not exactly define the data because they did not regard this as a major issue and postponed it. After applying Project Dependencies and Schedule, all project managers described their dependencies in detail and discovered that the contracts to be administrated by these applications are very individual and cannot be standardized.

Context

You work in a company running multiple software development projects in parallel and you have to manage their definition, interrelations like joint objectives, dependencies, and risks. For each project a project management plan should be used.

Problem

You have to create a consolidated schedule for all projects to care about the adherence to their plans and so the timely completion of the respective application. This schedule has to consider especially inter-project dependencies, the critical path [SW69], and the relations of involved organizational units. To reduce complexity, you also want to define projects effectively. The schedule should also be used for a high level key resource management for all projects.

The key question in this context is: **Which tasks have to be performed to ensure an effective project definition, inter-project dependency identification, risk management, and portfolio-strategy alignment?**

The following forces influence the solution:

Dependency management: Should dependency management be very strict or laissez-fair?

Integrity: How can you ensure the completeness of identified dependencies and keep the needed effort slight?

Change: Should you manage changing project requirements and schedules continuously or by a one-time approach?

Level of detail: How many projects can be included and how much information is appropriate to ensure manageability?

Requirements complexity: How can projects be defined effective regarding individual requirements and the application landscape vision?

Human reputation: Does the constraint of public status reporting result in information withholding?

Solution

Project Dependencies and Schedule follows a seven-step approach. The steps *Project setup* and *Planning* form a setup phase, in which projects are defined and the concrete proceeding is scheduled. It is followed by an analysis phase, covering steps *Dependency identification* and *Business support migration*, which puts the projects in the portfolio context. The last

phase, covering the steps *Prioritization*, *Joint scheduling*, and *CPM recomputing*, is an implementation phase which applies insights of previously executed phases.

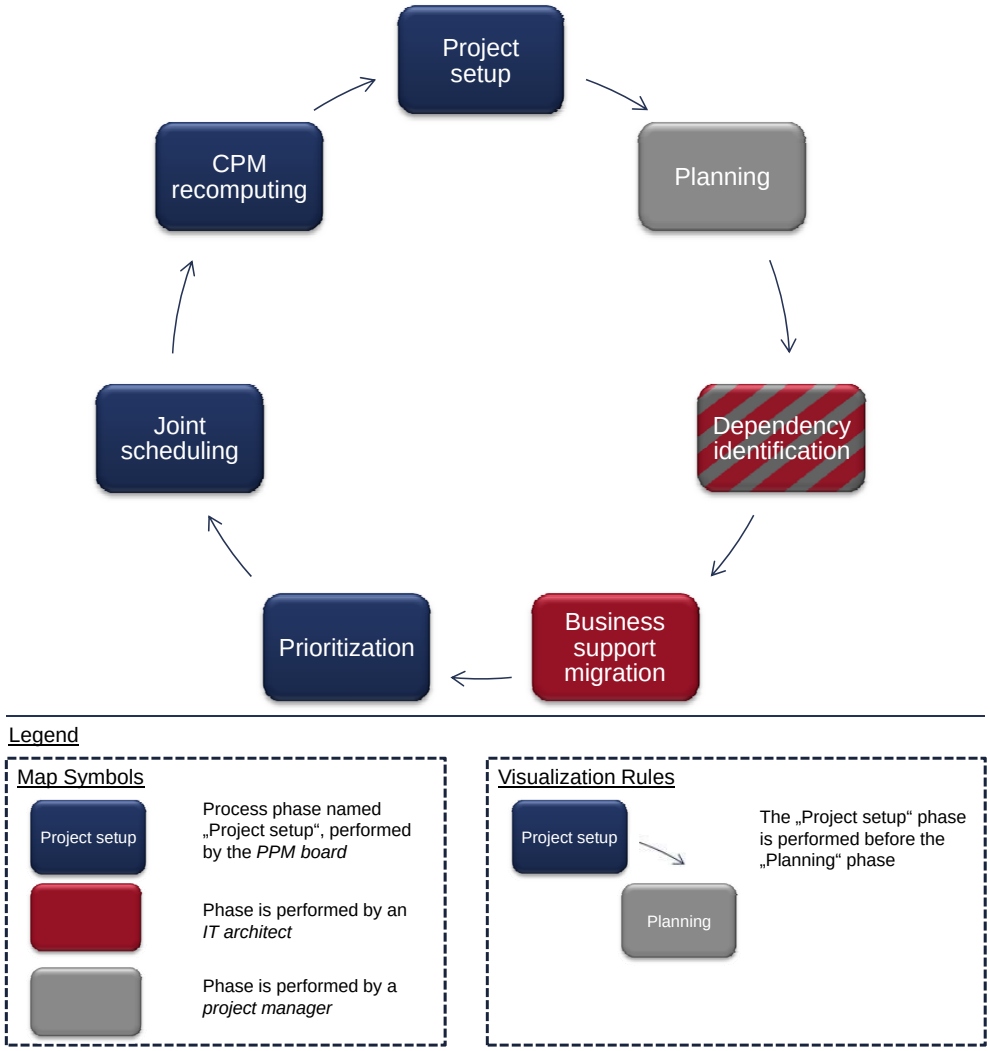


Figure 4.2: Project Dependencies and Schedule process

There are at least three conceivable events that are considered to occur likely and result in a new iteration of Project Dependencies and Schedule:

- *New requirements from business emerge* due to changing processes or new strategies.
- *Technical reasons* can require new applications or infrastructure like operating systems, e.g. if the vendor's time of support is over.

- *EA management decisions* can result in new projects, for example to consolidate the application landscape (see Variants section).

If one of the previously mentioned events occurred and a new iteration should begin all seven steps have to be performed. Here the respective steps are presented in detail:

Project setup During the project setup step the *project portfolio management board* has to define projects for the upcoming planning period. This might include budgeting, functional specification, and staffing. It is important, that not every request for a new application results in a single project. To address company-wide requirements, individual requirements need to be grouped to define appropriate projects considering the application landscape vision.

Planning During planning a project management plan is made for a previously set up project. This is done by the respective *project manager* and includes tasks like the creation of a work-break-down structure, estimation, and scheduling. For PROJECT DEPENDENCIES AND SCHEDULE this data is only needed on a summarizing level, but to get this data, the respective project manager will plan in detail.

Dependency identification This step is as essential as challenging. You will find some dependencies during the planning step, for example that test data is needed from another project. But there are also dependencies which are not that easy to identify. Therefore, this activity has to be done separately by *project managers* and *IT architects* together. At least, they should regard three types of dependencies:

- *Organizational dependencies:* They result from the organizational structure and context of the company. For example, two projects are going to roll out their application at the same organizational unit at the same time and both estimate that the rollout will take the whole weekend. The two IT employees at that organizational unit are not able to handle this. As a result, the two projects depend on one another because only one can rollout at this time. Another reason might be that the application which should deliver data to the project is not able to do this because it computes the annual accounts.
- *Interfaces between applications:* If a project uses data or functionality of another application currently developed, it depends on that projects ability to deliver.

- *Technical dependencies:* Usually, an application uses other software considered as infrastructure like, for example, databases or enterprise resource planning systems. A project depends on all other projects introducing or changing one of the used infrastructure systems.

In general, it is important, that both projects affected by a dependency formally commit to it. In addition, it is not enough to identify a dependency. They all have to be described in detail so everybody knows of all resulting impacts.

Business support migration This step might reveal additional dependencies and ensure the right handling of legacy systems. Therefore, it is necessary to determine all applications or parts of them which are retired by the currently developed application. If a legacy system can be retired immediately, the retirement process might also require another project. If only a part of the functionality is adopted, it has to be decided which application adopts the other parts if they are still needed. If some functionality is not needed any more, this fact also has to be documented.

Prioritization With the insights of the previous steps projects can now be prioritized relative to the other ongoing projects in the portfolio.

Joint schedule With the priority in mind, the new projects can now be included in the overall schedule (roadmap). If the new projects have any impact to other projects, these impacts have to be communicated.

Critical Path Method (CPM) recomputing After a roadmap update, the critical path needs be recomputed. The results might then have additional impacts to projects, which have to be applied and communicated.

Consequences

Through the systematical analysis of projects and their dependencies a better understanding of the project portfolio can be achieved. This requires a relatively strict dependency management including a formal process of commitment. The completeness of dependencies is important to keep risks at a minimum and efforts on an acceptable level. In order to perform the project setup step as proposed and handle complex requirements, a planning interval has to be determined. A longer interval allows better grouping

of requirements but will require more time until projects can be finished. If the time to market is the dominant success factor of projects a shorter interval should be determined. To ensure manageability and reduce level of detail of the process, only indispensable tasks are included. But to reach its targets the process has to be performed completely for all projects considered to have any dependency. Because the process depends on data about individual project schedules, it is important to convince project managers about their benefit of delivering this data and to prevent information withholding. If projects should be defined according to individual requirements satisfying individual divisions or according to a strictly consolidated application landscape vision is a matter of management style. Nevertheless, where requirement grouping is possible, the local autonomy needs to be constrained to keep the application landscape as simple as possible and take advantage of synergy effects.

Implementation

As already mentioned, different roles have to be assigned to people to perform the different steps of Project Dependencies and Schedule. In this context these roles can be defined as:

Project portfolio management board (PPM board) The project portfolio management board is responsible for tailoring and approving projects. They also prioritize projects according to the business or IT strategy and communicate rescheduling impacts.

Project manager A project manager is in charge for one or more projects of the project portfolio. He is responsible for the initial project data like phases and milestone dates.

IT architect The IT architect has the overview about the whole application landscape and is responsible for the consistent retirement of legacy systems and the adherence to the application landscape vision.

Additionally, the information collected during Project Dependencies and Schedule can be used to improve an overall risk management and also release management is directly supported with additional integrated information.

Variants

If the process starting event was an *EA management decision*, the **Business support migration** step will be performed at the beginning of the presented process. The insights are then a prerequisite to set up effective projects. In this case, the process steps will be performed in the following order:

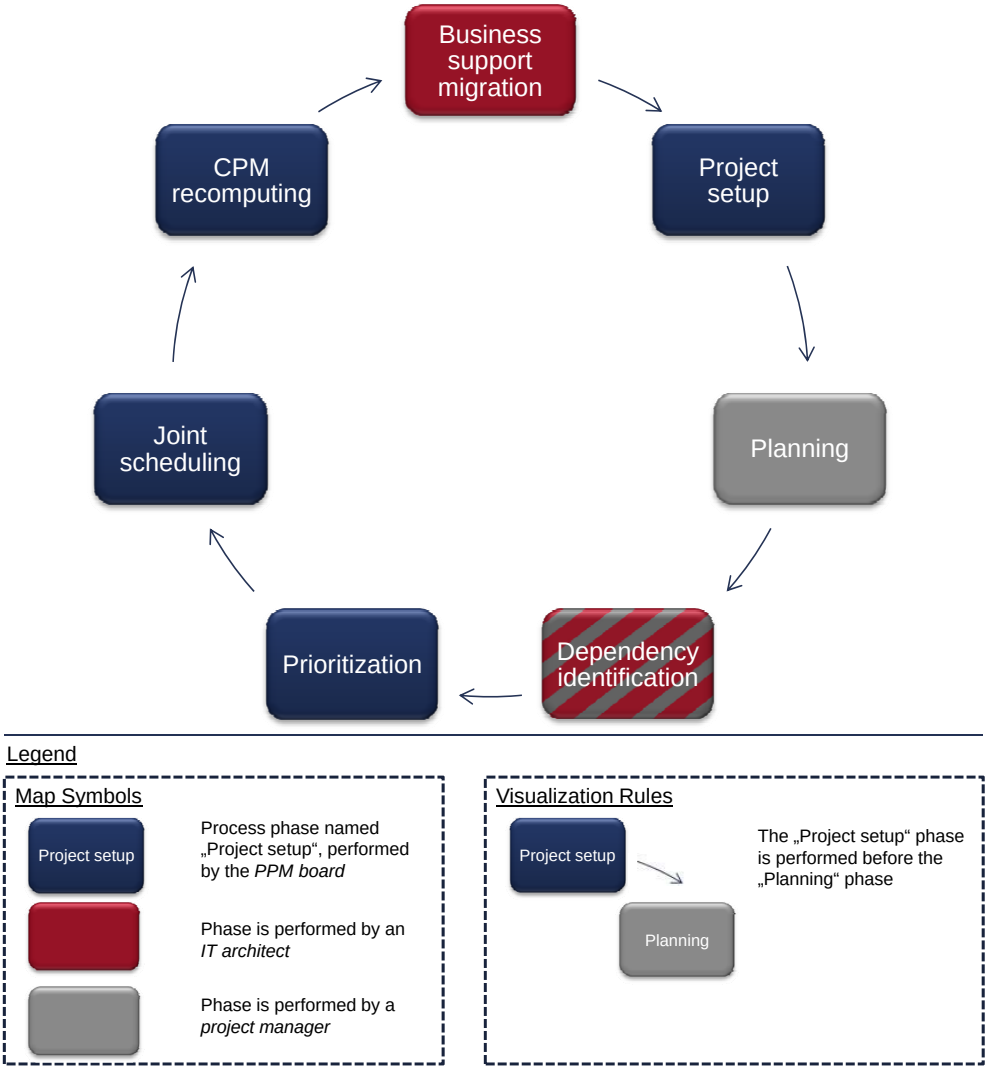


Figure 4.3: A variant of the Project Dependencies and Schedule process

Known Uses

Project Dependencies and Schedule is in use at Munich Re.

See Also

When using Project Dependencies and Schedule also the results of Project Portfolio Management and Monitoring of the Project Portfolio can be of interest because they are also concerned with projects and their relations. The implementation of this pattern can be supported using Project Portfolio Roadmap and Business Support Migration Map.

4.4 Project Portfolio Roadmap

Project Portfolio Roadmap provides a way to visualize information about projects, their progress, and their relations to each other. This includes project phases, milestones, and dependencies between projects.

Example

A Project Portfolio Roadmap was used to visualize dependencies between projects during the MRAm project conducted at Munich Re. It included about 20 projects and all their dependencies, relevant phases, and milestones. It was drawn with MS Excel, but quickly people requested a tool for generating the roadmap.

Context

In an enterprise, where multiple projects are conducted in parallel, an overview about all these projects is needed but hard to accomplish. Especially the interrelationships between these projects are often not consolidated. You might also search for a way to visualize such an overview to show it to other people not familiar with each project.

Problem

You want to display many projects at the same time and various information about each project. Projects can consist of sub-projects and have many milestones and phases, which can also consist of sub-phases. Between all these objects dependencies can exist and have to be visualized. You also need detailed information about these dependencies, which are not provided by common used PM tools.

The key question in this context is: **How can you visualize a project portfolio over time and include all project dependencies in a summarizing, comprehensive but detailed way?**

The following forces influence the solution:

Categorization: How can the great number of types and categories of phases, milestones, and dependencies be visualized without overcharging the user?

Change: How can you ensure that the roadmap always shows the latest information?

Format: How can the visualization be adjusted to the used medium?

Comprehensibility: Do acronyms simplify or complicate the visualization?

Solution

Figure 4.4 shows a view according to the respective viewpoint, which is inspired by the *Gantt-chart* and provides a Cartesian map with an x-axis made up by a time line with a monthly scale. The y-axis enumerates the projects of the current project portfolio and their related organizational units. The entities on the axes partition the main area of the visualization in cells. These intersections are filled with a symbol representing a project phase or milestone (roadmap objects), which expresses that the respective object is related to this project and takes place at this time. To provide more details, the exact day of the month is always added to each start or end of a phase or dependency. Dependencies between roadmap objects are shown as arrows. The arrowhead points on the object which depends on the other. Different types of dependencies are visualized by different arrow colors. In each project, several organizational units might be involved. If a roadmap object takes place at a certain organizational unit, it only appears in that row. The global row (as shown at Project G) summarizes the roadmap objects of all organizational units. If a project consists of sub-projects, the row for the encompassing project summarizes roadmap objects of its sub-projects the same way. In case there are also sub-phases defined for a project phase, they can also be shown (Phases 5, P5, P6 of Project F). Such an aggregation is only possible for phases of the same category, e.g. *testing*.

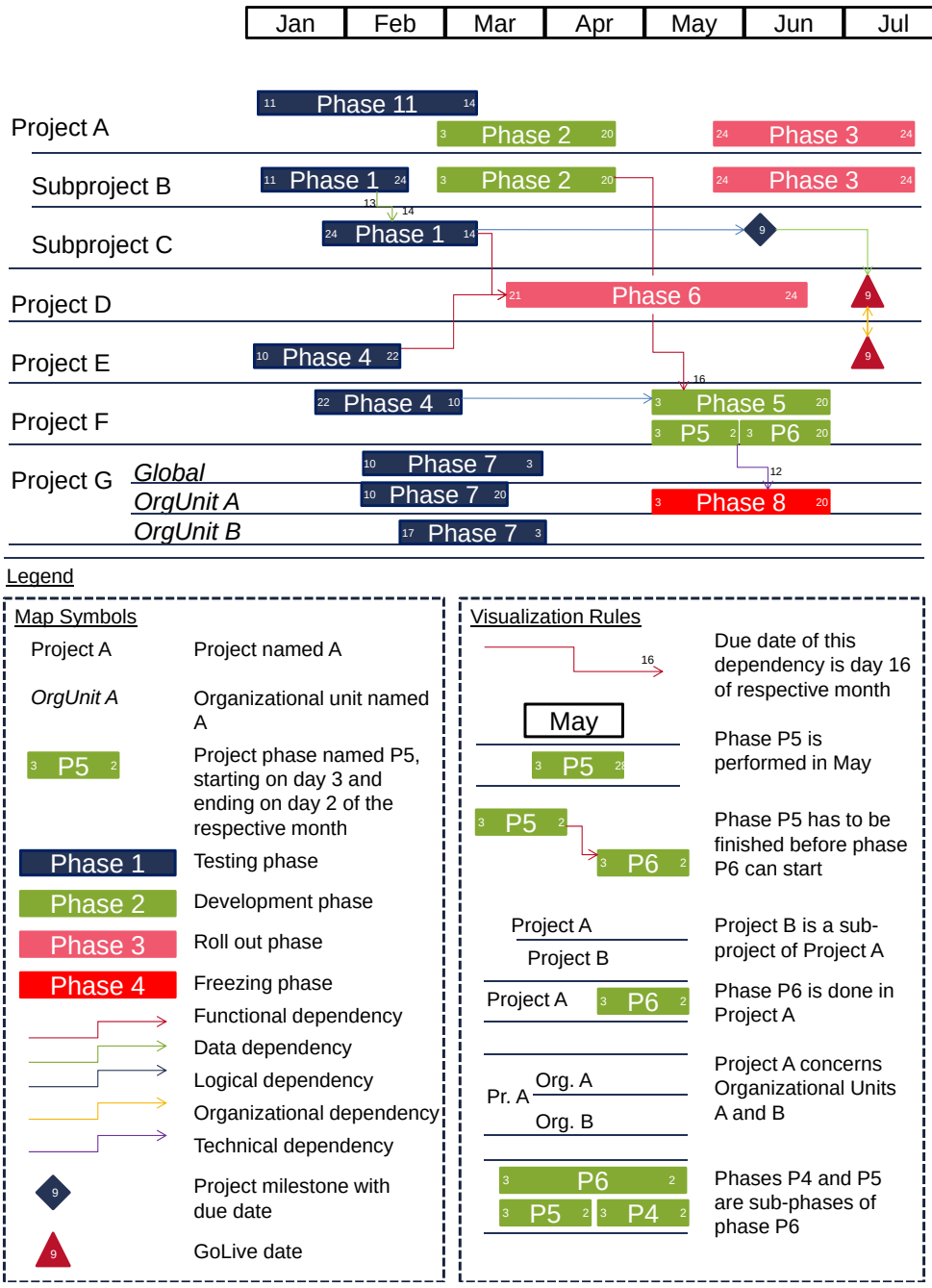


Figure 4.4: Exemplary Project Roadmap view

Implementation

Views according to this viewpoint can be created manually by any drawing tool, like e.g. Microsoft Excel. As manual creation is time consuming and error prone it is advised to use a tool, which can automate the creation of

such visualizations. There exist tools to create similar visualizations, like for example MS Project. But they are not able to categorize dependencies and add additional information about them. If the PROJECT PORTFOLIO ROADMAP should always show the latest information, automatic generation is recommended. In addition, different types of dependencies have to be determined by each company individually.

Variants

Another layout is to order the y-axis by organizational units first and then by projects. This might be appropriate, if for example all rollouts at a single organizational unit are of interest.

Consequences

To ensure the ability to display or print the PROJECT PORTFOLIO ROADMAP, appropriate filters showing only relevant objects have to be defined. In addition, project managers have to communicate every change in their project plan impacting the PROJECT PORTFOLIO ROADMAP. In order to create roadmap versions enduring an appropriate amount of time, changes should be allowed only at appropriate intervals. These intervals should be at least one week and should not exceed a month. Different categories of objects included are visualized by different colors. This is only possible for about six different types of each object because otherwise the user might be confused. The use of acronyms for project or phase names is not recommended, although it might be sensible to reduce needed space for printing enhancement. In this case a legend is strictly required.

Known Uses

PROJECT PORTFOLIO ROADMAP is in use at Munich Re.

MS Project also provides comparably views, but it is not able to categorize dependencies and add information about them.

See Also

Project Portfolio Roadmap is useful when using Project Dependencies and Schedule. The visualized information is based on Project Dependencies Representation.Comparable views, visualizing other subjects than projects, can be found in: Application Lifecycle Project Layer (V-27), Time Interval Map visualizing Projects and the affected Business Application (V-33), and Migration of Functionality (V-40).

4.5 Business Support Migration Map

Business Support Migration Map provides a way to visualize which application will take over functionality of other applications.

Example

The MRAm project was considered to roll out newly developed or adapted applications at a specific organizational unit. This was a subsidiary and therefore it used its custom built systems. In fact, this organizational unit had a very heterogeneous application landscape completely different from the parent company's. The goal was to replace many of the subsidiary's applications by these three globally used applications. To provide an overview for the management, which application will be replaced by which application Business Support Migration Map was used. The map visualized about 30 globally used applications replacing about 50 legacy systems.

Context

You work in a company which has many applications in use. For several reasons, new applications are developed, old applications are retired, and some functionality will be migrated from one application to the other. In order to get an overview, you need to visualize all introductions, migrations, and retirements affecting your application landscape.

Problem

Due to changing business processes and new technologies, applications currently in use have to be replaced by newly developed applications. Usually, this replacement is not done one-by-one. Instead, old applications can be retired by multiple new applications or many old applications can be retired by one new application. In order to understand this complex system of business support migrations and retirements, a comprehensible overview is needed.

The key question in this context is: **How can you visualize the replacement of old applications by new applications in a summarizing, comprehensive but detailed way?**

The following forces influence the solution of this problem:

Complexity: How can introductions, migrations, and retirements be visualized simultaneously?

Symbols: What are intuitive symbols and how can they be used to simplify the visualization without using too many different?

Format: How can you visualize a huge amount of applications, their relations, and additional information without exceeding a usable format?

Solution

To visualize information about introductions, migrations, and retirements of applications on a high level, a matrix representation can be used (Figure 4.5). The x-axis enumerates all applications which will hand over functionality to another application. The y-axis enumerates all currently planned or developed applications. If an application carries out functionality of another, a circle is put at the appropriate intersection in the matrix. According to the relative amount of functionality which will be overtaken, the circle is filled. A completely filled circle is used if the functionality is overtaken totally. The circle is half filled if about one half of the functionality is overtaken. The same applies to a quarterly and a three-quarterly filled circle. If the circle is empty, the amount could not be determined yet. If it is unclear, if an application adopts functionality of another application, but it is supposed to, a question mark is used instead of a circle. The last row of the matrix is used to visualize final retirements of respective applications listed on the x-axis, by showing a `X` and a date. The last column is used to visualize initial go-live events of applications listed on the y-axis. If the respective application is still in development, a triangle with a concrete date is shown to indicate the introduction.

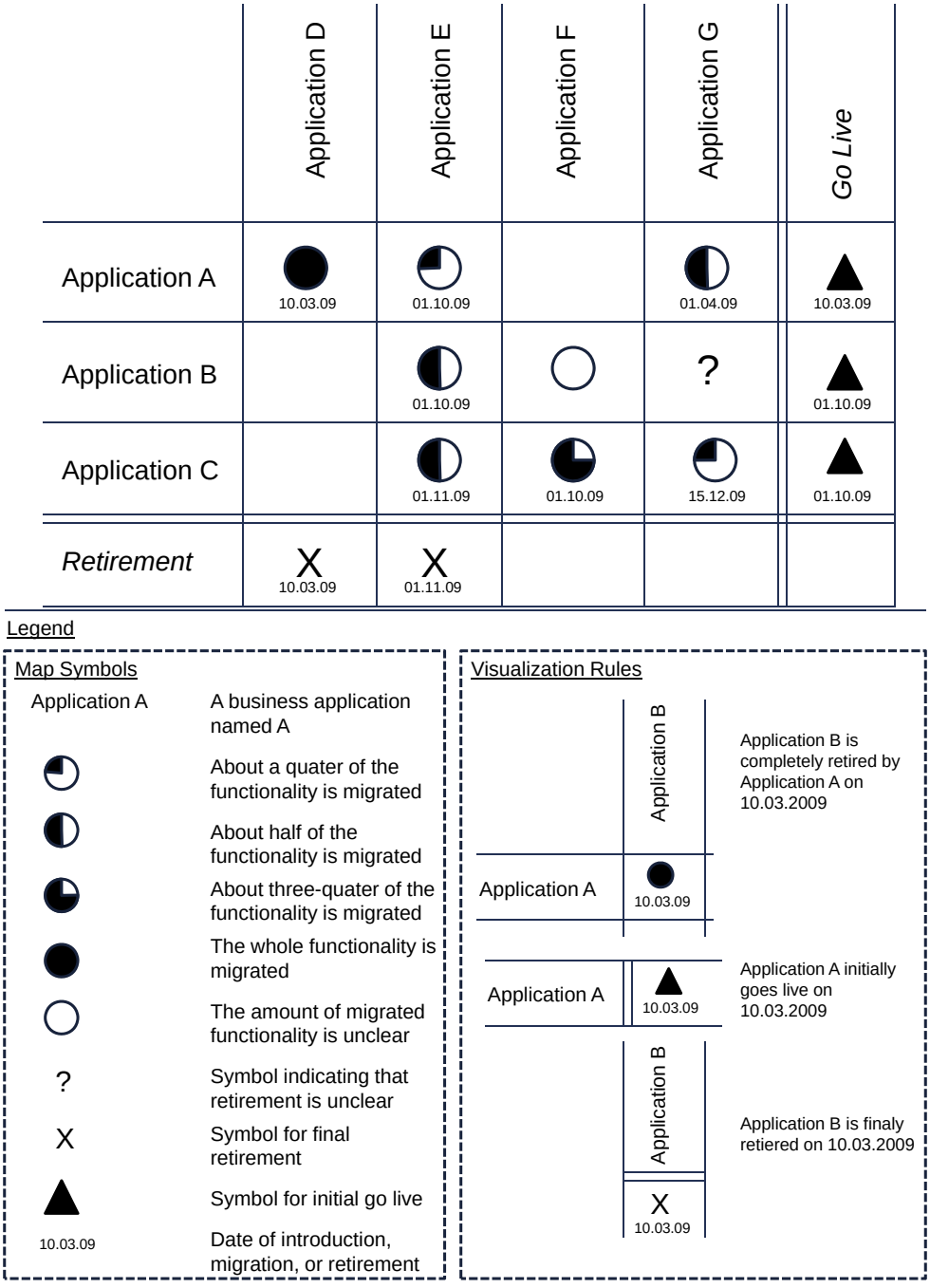


Figure 4.5: Exemplary Business Support Migration Map view

Implementation

Views according to this viewpoint can be created manually by any drawing tool like Microsoft Excel. As manual creation is time consuming and error

prone it is advised to use a tool, which can automate the creation of such visualizations. If Business Support Migration Map should always show the latest information, automatic generation is recommended. Printed on a large paper, it can be used as a baseline for group discussions and easily be filled manually.

Variants

Another layout possibility is a graph representation as shown in Figure 4.6.

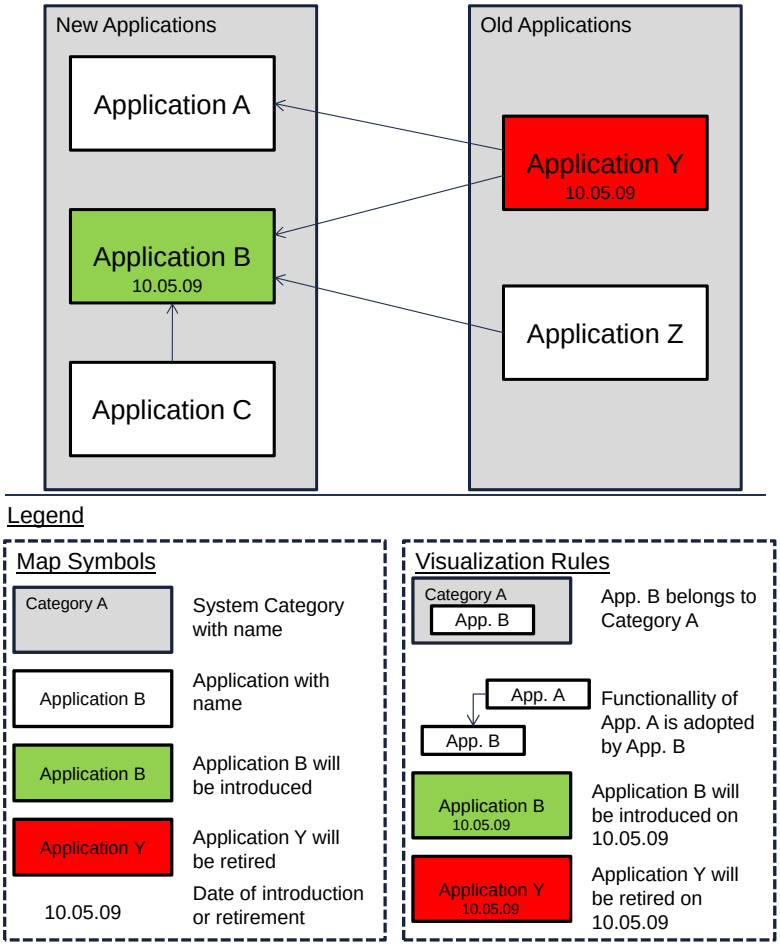


Figure 4.6: Variant of exemplary Business Support Migration Map view

Consequences

The Business Support Migration Map also provides assistance during the planning process of retirements by giving all legacy systems and the demand to specify the amount of functionality which will be retired. This visualization helps to ensure, that every part of a legacy system will be retired and that it really can be disabled without requiring new projects. Because usually people are interested only in applications used at one organizational unit, a respective filter should be provided to reduce complexity and keep the matrix small and straightforward. Additional information about functionality or application details should not be provided in order keep the matrix format small. Furthermore, no additional symbols should be used because such an overview visualization should be understandable for every user without intensively studying a legend. Introductions, migrations, and retirements are visualized separately but next to each other. This way of visualization helps the user to focus on the area he is interested in but also provides the whole information at once.

Known Uses

Retirement Map is in use at Munich Re.

See Also

Business Support Migration Map can help you to perform Project Dependencies and Scheduleand relies on data described in Business Support Migration. Comparable viewpoints are Process Support Map visualizing Changes in Relations to their Time Horizon (V-32), Overview over Lifecycle of Business Applications (V-36), and Migration of Functionality (V-40).

4.6 Project Dependencies Representation

Project Dependencies Representation provides a structure for organizing information about projects and their interrelationships.

Context

You work in a company conducting multiple projects in parallel and you want to collect data about these projects and their dependencies.

Problem

You want to store information about your different projects, especially about their dependencies and their schedules. Therefore, different project plans should be integrated in a project portfolio. The key question is: **"What is a good way to store and maintain information about IT projects?"**

The following forces influence the solution:

Data collection: Should the data be collected in intervals or should it be provided immediately after a change occurs?

Minimum effort: How can the effort to document the relationships between projects be minimized?

Solution

In order to integrate and consolidate different project plans, they need to be persisted at one single location. The conceptual UML class diagram shown in Figure 4.7 provides an information model structuring the important information needed to persist data about projects, their related organizational units, milestones, and dependencies. To represent the hierarchy of projects sub-projects can be grouped together. Each project is linked to its milestones and phases, which are summarized as `DependendObjects`.

For each `DependendObject` all organizational units where it takes place have to be determined and associated. To represent dependencies between `DependendObjects` the class `ProjectObjectDependency` always connects two of them and stores additional information about the reason for the dependency and its status. Because project phases can also consist of subphases they can be grouped together, too. Special milestones are so called go-live milestones. They indicate the point in time a respective business application starts operating. A business application can have more than one `GoLiveMilestone` because one is needed for every affected organizational unit. In order to provide a contact person or information about a respective project manager an employee is associated via a role to every project and `DependendObject`. Furthermore, external constraints are modeled. They also consist of `DependendObjects` which can be used to define dependencies from projects. A special type of external constraints are closing times which might occur in a large number of companies. During the closing time the associated business applications and the data they use might not be touched because they compute the annual accounts. For further information about EA evolution, see the Variants section.

The entities and relationships are defined in alphabetical order as follows:

BusinessApplication: A software system, which is part of an information system within an organization. An information system is therein according to [Krc05] understood as a socio-technological system composed of a software system (i.e. the business application), an infrastructure, and a social component, namely the employees working with the system. An information system is further described as contributing to the business process support demanded by the organization.

Closing: Is the time a `BusinessApplication` is computing the annual accounts. During this time, the application and/or the used data must not be touched.

ConcreteProject: A `Project` that is really performed.

DependencyCategory: An enumeration of categories a `ProjectObjectDependency` can belong to.

DependencyStatus: An enumeration of different states a `ProjectObjectDependency` can be in.

DependencyType: An enumeration of different types of

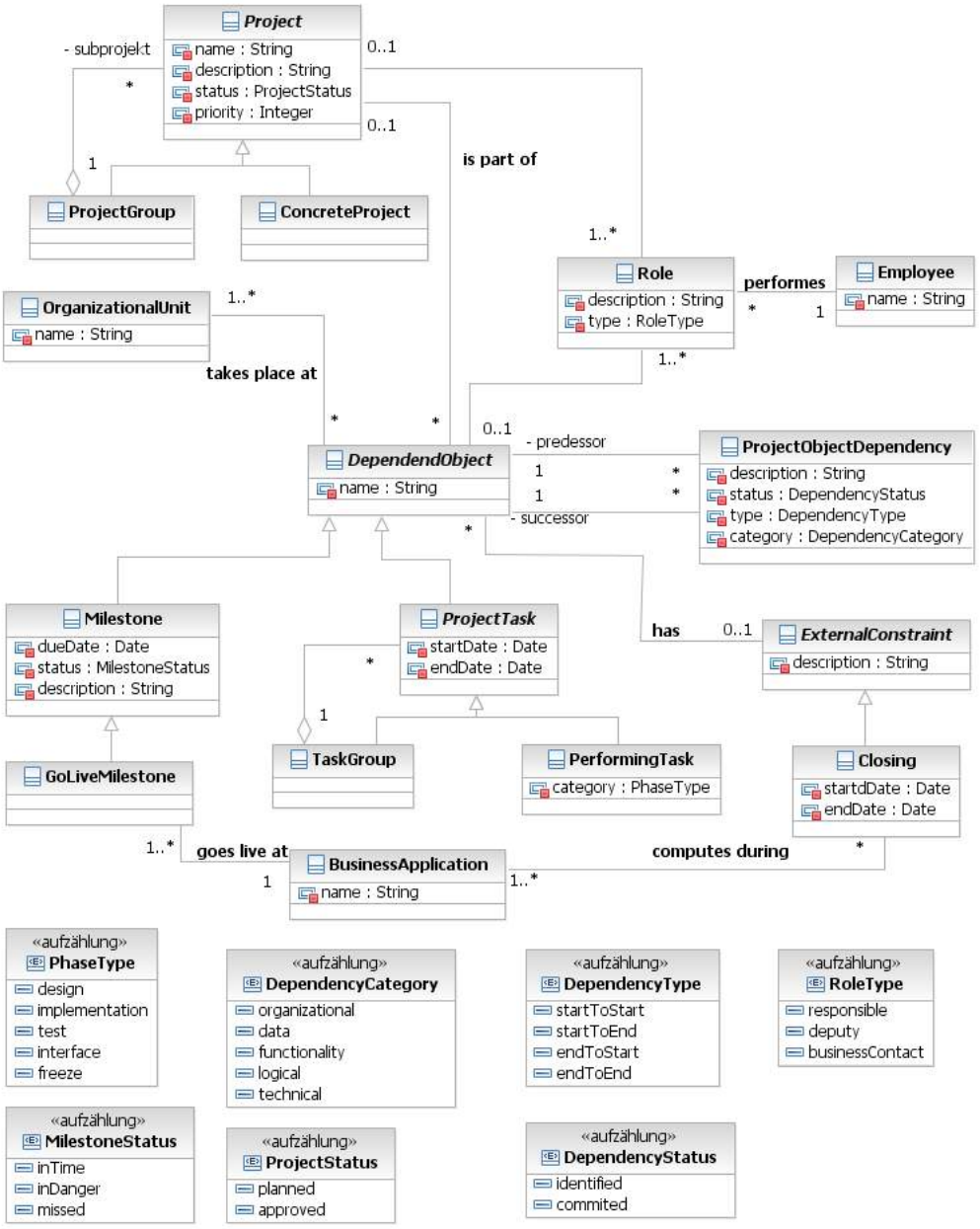


Figure 4.7: Project Dependencies Representation as conceptual UML class diagram

`ProjectObjectDependencies` indicating the timely order of corresponding `DependendObjects`.

DependendObject: A super-class for all objects which can depend on each other.

Employee: A person working for the company.

ExternalConstraint: Is an abstract super-class for all objects outside `Projects` but with direct impact to them.

GoLiveMilestone: A special `Milestone` indicating the first operational day of the developed application.

Milestone: A milestone marks defined points during the execution of a `Project`, where certain project activities should be completed. Thus, the progress of the `Project` can be measured. At a `Milestone` a certain effect on a `BusinessApplication` may occur.

MilestoneStatus: An enumeration of different states a `Milestone` can be in.

OrganizationalUnit: An organizational unit represents a subdivision of the organization according to its internal structure. A possible example are the entities showing up in an organization chart.

PerformingTask: A task performed during a `Project`, e.g. a project phase.

PhaseType: An enumeration of types a `PerformingTask` can belong to.

Project: A planned activity concerned with modifying one or more elements from the application landscape, mostly focused on `BusinessApplications`. Projects transform the application landscape.

ProjectGroup: A `Project` consisting of several sub-projects (`ConcreteProjects`).

ProjectObjectDependency: Stores information about a dependency between two `DependendObjects`.

ProjectStatus: An enumeration of different states a project can be in.

ProjectTask: Something that is done during a project, e.g. a project phase. This super-class is used to provide grouping of `ProjectTasks` via the composite pattern.

Role: This class associates an `Employee` to a `Project` or a `DependendObject`. *Type* indicates what role the `Employee` fulfills.

RoleType: An enumeration of roles an `Employee` can perform, e.g. in a `Project`.

TaskGroup: A `ProjectTask` which consists of several sub-tasks (`PerformingTask`).

BusinessApplication computes during Closing: Each `BusinessApplication` can have several time intervals for computing the annual accounts. During this interval it must not be touched.

BusinessApplication goes live at GoLiveMilestone: Each `BusinessApplication` has one `GoLiveMilestone` for each `OrganizationalUnit` indicating the beginning of its operating time.

DependendObject is part of Project: If an `DependendObject` is performed during a `Project` this association links them together.

DependendObject takes place at OrganizationalUnit: This association is used to indicate which `OrganizationalUnits` are directly affected by a `DependendObject`.

Employee is performs Role: Each `Employee` can have one or more `Roles` he can accomplish.

ExternalConstraint has DependendObject: Each `ExternalConstraint` can have several `DependendObjects` which can be used to define dependencies between the `ExternalConstraint` and `Projects`.

Implementation

Project Dependency Representation should be implemented in some kind of database system, because this is the easiest way to ensure the consistency of the relative high number of classes and associations. To ensure currentness of the information, data should be provided by project managers instead of asking for it in specific time intervals.

Consequences

When using Project Dependencies Representation, the amount of collected data is quite large because for all currently running projects many details need to be collected. To keep the effort as slight as possible only required information are shown in this pattern. The data about each project

itself might be easy to collect because each project manager has to create and store it anyway for his own purpose. More difficult would be the collection of data about dependencies between projects. For each kind of dependency (organizational, data, functionality, etc.) a corresponding specialist and all involved project managers have to work together. Beside the amount of information and its collection, the ongoing maintenance of the data caused by a usually high rate of change in project plans, is another challenge. Therefore, the interval of data collection should be at least a week. Furthermore, the planning of projects is supported by this pattern. If all necessary data is collected at the time of a project proposal, estimations about effort, time, and risk might be more precise. A possible further development of this pattern could be the extension to model information about specific resources, like for example people, to support high level resource management.

Variants

If `DependendObjects` like `Milestones` are considered to be part of more than one `Project` and can take place at different `OrganizationalUnits` in different `Projects`, a ternary relationship class associating `Projects`, `OrganizationalUnits`, and `DependendObjects` has to be used. For example one might define a `Milestone` for the end of all testing activities of several `Projects`. The testing takes place in `OrganizationalUnits` A and B in `Project` X and in `OrganizationalUnits` B and C in `Project` Y. If the `OrganizationalUnit` B should be removed from the testing `Milestone` but only for `Project` X, it will also be removed for `Project` Y if it is directly linked to the `Milestone` and not via a ternary relationship class.

Known Uses

Project Dependencies Representation is in use at Munich Re.

See Also

For visualizations of the included data Project Portfolio Roadmap and Business Support Migration Map might be considered. A process

determining how to collect the needed data can be found in Project Dependencies and Schedule.

4.7 Business Support Migration (Version 2.0)

Business Support Migration provides a structure for organizing information about the EA evolution including the introduction and retirement of applications and the migration of business supports.

Context

You work in a company which introduces new applications and retires old applications and you want to collect data about these relationships and the migration of respective business supports.

Problem

The key question is: **"What is a good way to store and maintain information about the EA evolution focusing application landscape evolution?"**

The following forces influence the solution:

Change log: How can you document changes of an Enterprise Architecture with minimal effort?

Level of detail: What information about migrations is additionally needed and can be easily collected?

Solution

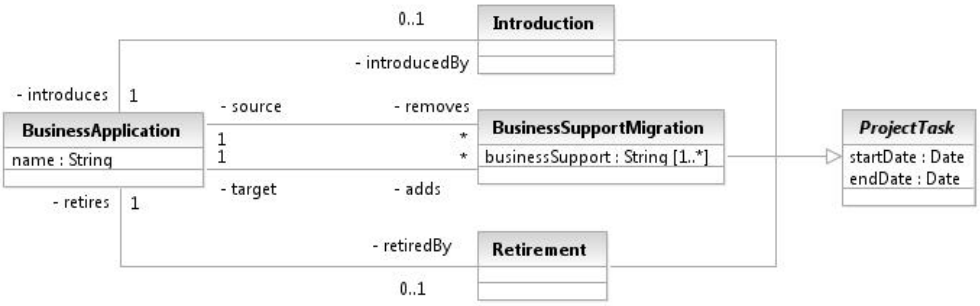


Figure 4.8: Business Support Migration as conceptual UML class diagram

Business Support Migration consists of the entities Business Application, Business Support Migration, Project Task, Introduction and Retirement. They are defined as follows:

Business Application is a software system, which is part of a business information system of an organization. A business application thus provides support for at least one business process, i.e. infrastructure systems are not considered business applications in this context.

Business Support Migration represents a project task migrating the provision of a specific business support from a *source* business application to a *target* one. The business support is considered fully migrated, once the date specified in *endDate* has passed.

Introduction is a specific type of project task introducing a distinct `BusinessApplication`. After the date specified in *endDate*, the associated `BusinessApplication` is considered to be in production.

Project Task is the abstract base concept for the different accomplishments of projects as considered in this pattern. Each project task spans a distinct period of time, enclosed by the two points in time *startDate* and *endDate*. The project tasks indicate the discrete events of change, connecting the different states of the EA to a chronological sequence.

Retirement is a specific type of project task retiring a distinct `BusinessApplication`. After the date specified in *startDate*, the associated `BusinessApplication` is considered to be in retirement.

The information model fragment is complemented with OCL constraints, which are used to enforce complex consistency requirements that have to hold in respective instance models. The first constraint imposes, that no project task may end, before it has started.

```
context ProjectTask
inv:  startsAt ≤ endsAt
```

A second constraint applies, demanding that a business support is always migrated between different business applications, i.e. that the source and target application of the respective migration must not be identical.

```
context ProjectSupportMigration
inv:  source ≠ target
```

A third constraint concludes the consistency conditions, by imposing, that the respective source and target business applications of a business support migration must exist, i.e. they must be at least under development or in retirement.

```
context ProjectSupportMigration
inv:  source.intorducedBy = null ∨
(startsAt ≥ source.intorducedBy.startsAt)
inv:  source.retiredBy = null ∨
(startsAt < source.retiredBy.endsAt)
inv:  target.introducedBy = null ∨
(endsAt ≥ target.introducedBy.startsAt)
inv:  target.retiredBy = null ∨
(endsAt < target.retiredBy.endsAt)
```

The above constraints additionally consider the absence of an introducing or retiring project task, i.e. if none such task is associated, the respective business application is considered to have been in production ever since or remain in production ever after. Especially the first case may appear in practice, as perhaps the introducing project task of a business application has not been modeled.

Implementation

Business Support Migration should be implemented in some kind of database system, because this is the easiest way to ensure the consistency constraints mentioned in the solution.

Consequences

Based on the information stored according to the model introduced above, not only roadmaps of EA transformation can be created - it is further possible to create comparisons of two EAs (or cutouts thereof), indicating the architectural delta between the state before and after the execution of a distinct sets of project tasks. This can be achieved as the sequence of project tasks forms a kind of discrete change log of an evolving EA. The amount of functionality added to the `BusinessSupportMigration` in the variant provides a better understanding of all migrations. Because the determination of the exact amount of migrated functionality might be difficult only an estimation is necessary.

Variants

Figure 4.9 shows a variant of Business Support Migration. The `BusinessSupportMigration` class now has an additional attribute and an enumeration class:

functionalityExtend: Indicates the amount of functionality which will be migrated from *source* to *target*.

Measure: An enumeration of functionality extends.

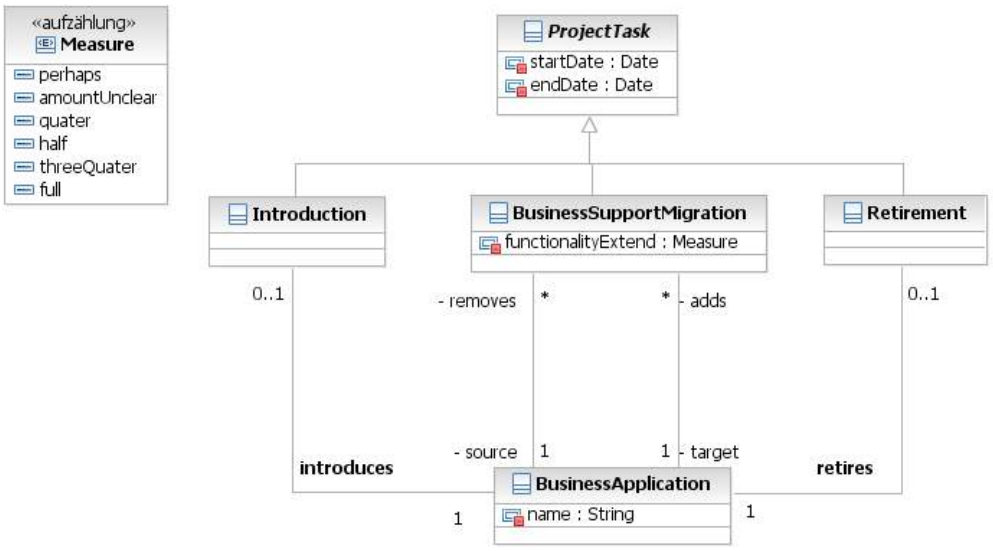


Figure 4.9: Variant of Business Support Migration as conceptual UML class diagram

Known Uses

Business Support Migration is in use at Munich Re.

See Also

Business Support Migration is the basis for Business Support Migration Map.

4.8 Assessment of the Pattern-Based Approach to EA Management

The pattern-based approach to EA management was used throughout the conduction of this thesis. The following paragraphs determine advantages and disadvantages relevant for the author during the different phases of this thesis:

Literature and current approach analysis

Benefits: During the analysis of Munich Re's current approach to EA evolution management the different types of EAM Patterns provided a structure for the procedure. The analysis first focused on the stakeholders and the process performed because these topics are normally provided by M-Patterns. According to the purpose of V-Patterns the second step was to analyze the way of visualization. At this point especially the distinction between view and viewpoint [IEE07] helped to understand the way objects were visualized. Subsequently, all information used was analyzed regarding especially the relationships between different objects which is usually done in I-Patterns. As a result, the insights could directly be used for further proceeding. The V- and I-Patterns documented in this chapter directly emerge from the insights and the documentation of the previous executed analysis.

Requirements elicitation

Benefits: Also during the interviews performed during requirements elicitation the distinction of EAM Pattern types was helpful because requirements could be categorized according to the pattern type they concern. Furthermore, the interview proceeding was also influenced because it was easier to determine whether people ask for a new process, new information or just for another way of visualization. After that the interviewer was able to go into such a requirement in detail.

EAM Pattern documentation

Benefits: During the documentation of the insights achieved in previous phases of this thesis the fixed structure of EAM Patterns ensured

a complete documentation of all available knowledge. Everything needed to be documented fits in one of the several EAM Pattern sections and almost all sections were filled.

Liabilities: EAM Pattern phrasing differs from common phrasing used in academic publications. It is less formal and directly addresses the reader. To find the appropriate phrasing was sometimes difficult. For practical users the casual phrasing might be a benefit because the content is easy to access.

Conclusion

In summary, the used pattern-based approach to EA management is useful to achieve the targets of this thesis and is also useful for the company supporting this thesis. Because the documented EAM Patterns are related to each other, but in fact completely independent, they can be implemented step-by-step. Because they address new concerns they can also be included in the *EAM Pattern Catalog*.

Chapter 5

Conclusion and Outlook

Finally, this chapter summarizes the results of this thesis. Afterwards, an outlook for further research topics about the management of EA evolution is provided.

5.1 Conclusion

The target of this thesis was to analyze and document Munich Re's approach to EA evolution management. After a literature inquiry, their current approach was reverse engineered. Further requirements were collected by interviews with several stakeholders and users. The pattern-based approach to EA management was then used to document the insights of the analysis phase and the requirements elicitation. The resulting EAM Patterns provide a process, visualizations, and information models to face the challenge of EA evolution management.

Initially, in Chapter 2.1 a literature analysis of relevant disciplines was conducted in order to achieve a broad understanding of EA evolution. Chapter 2.2 depicts the analysis of Munich Re's current approach to EA evolution management. It provides reverse engineered legends and information models to keep the sparsely documented approach manageable. On the basis of these chapters, an interview guideline, which is shown in Appendix A, was developed to be used during the interviews for additional requirements collection. Chapters 3.10 and 3.11 compare the raised requirements with the current approach and restrict the focus to important requirements which will be addressed by EAM Patterns. These EAM Patterns are docu-

mented in Chapter 4. Project Dependencies and Schedule provides a process and defines roles for accomplishing EA evolution management. Project Portfolio Roadmap and Business Support Migration Map show visualizations, which might help for managing the EA evolution. Finally, Project Dependencies Representation and Business Support Migration show how information for EA evolution management can be stored.

Altogether, it can be concluded, that EA evolution management is still in early stages of development both in literature and practice. For the same reasons, why EA management is an important activity, also its evolution has to be managed. It is the only way to ensure the validity and manageability of once collected EA data for the future. It can be expected, that it will become a crucial part of every EA management approach.

5.2 Outlook

The EAM Patterns documented in this thesis provide a baseline for managing the EA evolution. Nevertheless, it is not a one-time approach and might be extended. For example, the ability of scenario creation or simulation of changes is currently not provided, although there was such a requirement within the reinsurance company. To support scenario creation, the integration of different timelines and versions in an I-Pattern is necessary, which can be much more complex than it seems at the first glance [BEMS09]. Furthermore, the collected and visualized information can be more detailed. Currently, the documented EAM Patterns do not visualize or contain information about infrastructure systems or the mapping of business applications to infrastructure systems they use. In addition, the proposed states of objects in Project Dependency Representation are not complete and might be extended. It might also be examined, how the newly documented EAM Patterns work together with existing patterns of the *EAM Pattern Catalog* also concerning project portfolio management or the evolution of the EA. Last but not least, the documented patterns are in use at only one company. In order to provide proven practice solutions, the patterns need to be validated by other companies.

Bibliography

[AG09] Australian Government Department of the Prime Minister and Cabinet. Guidelines for new policy proposals. `http://www.pmc.gov.au/implementation/policy.cfm`, May 2009.

[ARW08] Stephan Aier, Christian Riege and Robert Winter. Unternehmensarchitektur: Literaturüberblick und Stand der Praxis. *Wirtschaftsinformatik*, 50:292–304, August 2008.

[Bac96] David Baccarini. The concept of project complexity - a review. *International Journal of Project Management*, 14:201–204, August 1996.

[BBB+01] Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. Manifesto for agile software development. `http://agilemanifesto.org/`, 2001.

[BD04] Bernd Brügge and Allen H. Dutoit. *Object oriented software engineering.* Pearson Prentice Hall, 2. edition, 2004.

[BEMS09] Sabine Buckl, Alexander M. Ernst, Florian Matthes, and Christian M. Schweda. An information model for managed application landscape evolution. *Journal of Enterprise Architecture*, pages 1–15, 2009.

[BLM05] Alexander Bogner, Beate Littig, and Wolfgang Menz. *Das Experteninterview: Theorie, Methode, Anwendung.* 2. edition, 2005.

[Bro86] Frederick P. Brooks. No silver bullet: Essence and accidents of software engineering. *Information Processing*, 1986.

[BT76] Thomas E. Bell and T. A. Thayer. Software requirements: Are they really a problem? 1976.

[CA09] British Educational Communications and Technology Agency. Release management. `http://www.becta.org.uk`, 2009.

[CC90] Elliot J. Chikofsky and James H. Cross. Reverse engineering and design recovery: A taxonomy. *IEEE Software*, pages 13–17, 1990.

[Cou09] Liam Coughlan. Project Management. `http://www.acca.com.ge/projectmgt.htm`, May 2009.

[DLP02] A. DeMeyer, C.H. Loch, and M.T. Pich. A framework for project management under uncertainty. *Sloan Management Review*, 43:60–67, 2002.

[Dun96] W.R. Duncan. A guide to the project management body of knowledge, 1996.

[Ern07] Alexander M. Ernst. *A pattern-based approach to Enterprise Architecture Management*. PhD thesis, Fakultät für Informatik, Technische Universität München, 2007. in publication.

[FV04] Guido Fioretti and Bauke Visser. A cognitive interpretation of organizational complexity. *Emergence: Complexity and Organization*, 6:11–23, 2004.

[GL93] Joseph A. Goguen and Charlotte Linde. Techniques for requirements elicitation. *Social Science*, 1993.

[GR04] Donald R. Greer and Günther Ruhe. Software release planning: an evolutionary and iterative approach. *Information and Software Technology*, 46:243–253, March 2004.

[HHHW97] Andre Van Der Hoek, Richard S. Hall, Dennis Heimbigner, and Alexander L. Wolf. Software release management. *ACM SIGSOFT Software Engineering Notes*, 22:159 – 175, 1997.

[IEE98] Institute of Electrical and Electronics Engineers. IEEE standard for software project management plans. 1998.

[IEE07] Institute of Electrical and Electronics Engineers. IEEE std 1471-2000: IEEE recommended practice for architectrale description of software-intensive systems. *IEEE Computer Society*, 2007:c1–24, 2007.

[Joh01] James H. Johnson. Micro projects cause constant change. *Proceedings of the Second International Conference on Extreme Programming and Agile Processes in Software Engineering*, pages 20–23, 2001.

[Kel07] Wolfgang Keller. *IT-Unternehmensarchitektur*. dpunkt, 1. edition, 2007.

[Luc05] Gunnar Lucko. Reviving a mechanistic view of CPM schedules in the age of information technology. *Proceedings of the 2005 Winter Simulation Conference*, 1533–1540.

[LW04] Karl Langenberg and Alain Wegmann. Enterprise architecture: What aspects is current research targeting? *EPFL Technical Report IC/2004/77*, 2004.

[Mar52] Harry Markowitz. Portfolio Selection. *The Journal of Finance*, 7:77–91, 1952.

[McF89] F. Warren McFarlan. *Portfolio approach to information systems*, pages 17–25. IEEE Press, Piscataway, 1989.

[MJS+00] Hausi A. Müller, Jens H. Jahnke, Dennis B. Smith, Margaret-Anne Storey, Scott R. Tilley, and Kenny Wong. Reverse engineering: A roadmap. *Proceedings the International Conference on Software Engineering (ICSE-00)*, pages 47–60, 2000.

[Mor97] Peter W. G. Morris. *The Management of Projects*. Morris, Peter W. G., 1997.

[Mun09] MunichRe. Munich Re Internal Website, 2009.

[MWT00] R. Murray-Webster and M. Thiry. *Project programme management*. Gower, London, 3 edition, 2000.

[Ois71] R.P. Oisen. Can project management be defined? *Project Management Quarterly*, 2:12–14, 1971.

[RA05] Annie I. Rana and Muhammad W. Arfi. Software release methodology: A case study. *2005 Student Conference on Engineering Sciences and Technology*, pages 1–10, August 2005.

[RGCL+05] Bert De Reyck, Yael Grushka-Cockayne, Martin Lockett, Sergio Ricardo Calderini, Maricio Moura, and Andew Sloper. The

impact of project portfolio management on information technology projects. *International Journal of Project Management*, 23:524–537, 2005.

[SOX02] U.S. Congress. Sarbanes-Oxley Act of 2002. *Pub. L. No. 107-204*, 2002.

[seb05] sebis. Enterprise Architecture Management Tool Survey 2005. *Technische Universität München, Chair for Informatics 19 (sebis)*, 2005.

[seb09a] sebis. EAM Pattern Catalog. `http://eampc-wiki.systemcartography.info/wikis/eam-pattern-catalog/home`, 2009.

[seb09b] sebis. SyCa - System Cartography. `http://wwwmatthes.in.tum.de/wikis/sebis/syca`, 2009.

[SGF01] Gary Stoneburner, Alice Goguen, and Alexis Feringa. Risk management guide for information technology systems. *Nist Special Publication*, 800-30, 2001.

[SW69] H.S. Swanson and R.E.D Woolsey. A pert-cph tutorial. pages 54–62, 1969.

[The09] The Open Group. Togaf version 9. `http://www.opengroup.org/togaf`, 2009.

[Tum86] J. Tuman. Success modeling: A technique for building a winning project team. *Measuring Success, Proceedings of the 18th Annual Seminar Symposium of the Project Management Institute*, pages 94–108, September 1986.

[Tur96] J.R. Turner. Editorial: International project management association global qualification, certification and accreditation. *International Journal of Project Management*, 14:1–6, 1996.

[Tur04] J.R. Turner. *The handbook of project-based management*. McGraw-Hill, 2 edition, 2004.

[Wal09] Ingrid Walter. EAM at Munich Re, May 2009.

[Wil95] Terry Williams. A classified bibliography of recent research relating to project risk management. *European Journal of Operational Research*, 85:18–38, August 1995.

[Zac99] John A. Zachman. A framework for information systems architecture. *IBM Systems Journal*, 38:454–470, 1999.

Appendix A

Interview guideline

This guideline was used as a basis during the expert interviews described in Chapter 3. It is composed of different modules which address the fact, that knowledge and experience differ widely among the interview partners. Before each interview, a specific guideline was assembled of these modules. The modules described in Sections A.2, A.3, A.5, A.6, A.8 and A.9 were included in each specific guideline. In the following, a list of all modules can be found.

A.1 Central question

Which requirements for a roadmap for enterprise architecture evolution exist within the enterprise?

A.2 Greeting and introduction

1. Say welcome to the interview partner.
2. Short introduction about this thesis.
3. Short introduction about the interview proceeding and duration.
4. Do you agree to an audio recording of this interview?
5. Did you have a look at the interview guideline I sent to you?

A.3 Questions about the person

1. What is your job title?
2. In which division do you work?
3. How long are you in this position?
4. What is your area of responsibility?

A.4 Questions about the current approach

1. Did you use the current roadmap?
2. In which information are you interested?
3. Were they crucial for your work?
4. Was the provided information sufficient? Which information was not included?
5. Is there anything you were bothered about?
6. Do you know somebody else, who used the current roadmap?

A.5 Questions about release and rollout management in general

1. Do you see a need for a roadmap for enterprise architecture evolution?
2. When do you need information about development and maintenance projects?
3. In which information are you interested in detail?
4. Which grade of detail is needed (i.e. projects, phases, etc.)?
5. Is a distinction between maintenance and development projects necessary?
6. Which kind of representation do you prefer?
7. How important is version control or access control?

A.6 Questions about retirements

1. Do you need information about the retirement of legacy systems?
2. Which information do you need in detail?
3. Should the retirement of legacy systems be included in the roadmap?
4. What grade of detail (which attributes) do you need?
5. Which kind of representation do you prefer?
6. Are you interested in a representation of the transitive closure?

A.7 Questions about the global portfolio management

1. Which goals does the GPM pursue?
2. Are there any requirements already documented?
3. Which information is going to be gathered?
4. Which visualizations are planned?
5. Which tools will be used?
6. Who are the expected users?

A.8 Further questions

1. Who in your opinion has the overview of all global projects and their dependencies? Who should have it?
2. Who else might be interested in a roadmap for enterprise architecture evolution?
3. Is there any question you have?

A.9 Farewell

1. Acknowledgment.
2. Are you interested in a copy of this thesis?