

Architecture-Driven Requirements Engineering

Wilco Engelsman¹, Henk Jonkers¹, Henry M. Franken¹, and Maria-Eugenia Iacob²

¹ BiZZdesign, P.O. Box 321, 7500 AH Enschede, the Netherlands

² University of Twente, School of Management and Governance, P.O. Box 217, 7500 AE Enschede, the Netherlands

Abstract. This paper presents an architecture driven requirements engineering method. We will demonstrate how to integrate requirements engineering in architecture design and we will demonstrate how to use enterprise architectures during solution realization projects. We will demonstrate how architecture can be used during problem investigation, solution specification and solution validation through an example application.

Keywords: Requirements Engineering, Goal Oriented Requirements Engineering, Enterprise Architecture.

1 Introduction

In the last few years the enterprise architecture (EA) paradigm emerged to better adapt organizations to changing customer needs [25] [9]. EA is believed to increase the understanding of scope for solution realization projects. Solution realization projects in this context are the projects executed to reach the to-be architecture, for example introducing a new business service.

Requirements Engineering (RE) as a scientific discipline has matured over the last decade. The process itself is rather well understood and has led to numerous techniques and models (e.g. GBRAM [1], I* [24] KAOS [19] and more traditional techniques like interviews, workshops [3] or viewpoint oriented RE [10]). However, we believe that enterprise architectures can have a tremendous impact on the requirements engineering process. Not only do requirements lead to architecture, there is also much progress to be made in constraining the requirements process with the relevant scope, context and structure. Lastly eliciting and specifying requirements from architectural models can give the requirements engineers a head start before traditional techniques like workshops and scenario based elicitation come into play. We believe that there is much progress to be made by clearly defining the relationships between RE and EA. To explore our ideas on the proposed integration of architecture into requirements engineering we performed an exploratory case study at a large Dutch insurance company to extract this information. In this paper we will demonstrate how requirements engineering leads to architecture and architecture leads to requirements. This distinction can be made because architecture is either a design artifact or a frame of reference. We will position requirements engineering in both these views and propose a way of integrating these views.

The structure of this paper is as follows, in section 2 we will demonstrate our view on requirements engineering. In sections 3 and 4 we provide the theoretical boundaries of architectures and RE and describe the conclusions from our study. In section 5 we provide a framework for architecture-driven requirements engineering and in section 6 we provide an example application of our method. In chapter 7 we provide an outlook for further research.

2 Requirements Engineering

Requirements Engineering (RE) is involved with investigating and describing the environment in which the envisioned system is supposed to create desired effects and designing and documenting behavior of the envisioned system [3]. In other words, RE is about getting from problems to the possible solutions. In general, there might be more than one valid or needed solution for a particular problem. Each solution itself can be another problem for someone else (see figure 2). This was recognized by Jackson as a progression of problems [7].

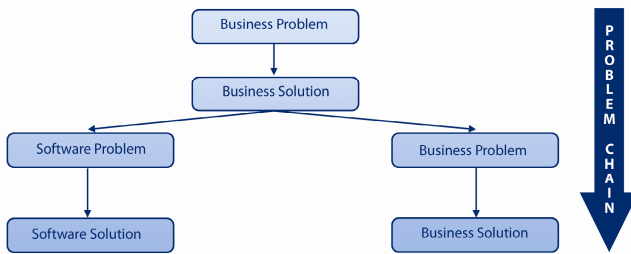


Fig. 1. A progression of problems

We define RE as getting from problems to the possible solutions. We do not limit ourselves to technology based solutions. Therefore we define a solution as a system that provides desired services. A system can be a new information service to customers, new business processes, new work procedures, supporting software systems and application services that support business processes.

Because RE is about bridging the gap between a problem and the possible solutions, two different views on RE have emerged [2] [21] [22]: *problem-oriented* and *solution-oriented* RE. Problem-oriented RE is about problem investigation: to investigate and determine what the actual problem is. Problem-oriented RE involves finding and documenting the problematic phenomena before thinking of how to solve that particular problem. A key concept in this is: understanding the goals and the stakeholders who experience these goals. Solution-oriented RE is about designing and describing system behavior and showing which alternative best solves the problem. If we try to relate these two views, we can argue that problem-oriented RE and solution-oriented RE come together in *architecture* [21]. If we investigate a certain problem, for example, we determined that certain stakeholders experience that the service delivery to customers is insufficient to realize certain business goals. Then a solution to this problem could be

new business processes, a new delivered service to the customer, alignment of existing business processes to this new service and a new supporting software system. Together these different solutions form the architecture of an overall solution (see figure 2). We will elaborate this statement in chapter 4 and 5.

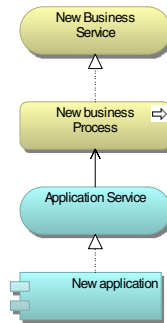


Fig. 2. Overall solution architecture

2.1 Problem-Oriented RE

In this section we look more closely at Requirements Engineering (RE) as a problem solving activity. This view on RE originated from systems engineering and is about investigating and documenting a problem domain. Within this view the requirements engineer describes the experienced problematic phenomena, the relations between these phenomena, why this is seen as problematic and who experiences these problems.

Wieringa [2] [21] provides us with information about what a Requirements Specification (RS) should contain when RE is seen as problem analysis; a RS in this view describes the desired business objectives and what work should be done to reach these business objectives. A similar distinction can be found in Tropos [4]. Tropos uses an early and late requirements phase, where the early requirements phase describes the system objectives and the late requirements phase describes the functional and non-functional requirements.

A very popular RE technique within problem-oriented RE is Goal Oriented RE (GORE). GORE [1] has received a large amount of research efforts over the past years and its popularity has increased ever since. Goals are regarded as high-level objectives of the business, organization or system. They capture the reasons why a system is needed and guide decisions at various levels within the enterprise. For a general description about GORE in practice see the work of Van Lamsweerde [18]. Relevant work in the field of GORE has been done by the authors of GBRAM [1], KAOS [19] and I* [24]. The main reason to adopt a GORE based approach is the inadequacy of traditional system approaches (e.g. structured analysis or object oriented analysis) to capture the actual motives for the system under development. Traditional approaches treat requirements as consisting only of data and processes and do not capture the rationale for the systems, making it difficult to understand high-level concerns in the problem domain.

2.2 Solution-Oriented RE

This view on requirements engineering (RE) is the traditional software engineering view on requirements engineering. When using the view of solution specification a requirements specification consists of [21]:

- A specification of the context in which the system will operate.
- A list of desired system functions of the system.
- A definition of the semantics of these functions.
- A list of quality attributes of those functions.
- A Demonstration which alternative best solves the problem.

Traditional techniques in this view are structured analysis [15] and object-oriented analysis [8]. Object-oriented analysis applies techniques for object-modeling to analyze the functional requirements for the system under development. Structured analysis focuses on the data that flows through the system under development.

3 Enterprise Architecture and Requirements Engineering

Enterprise Architecture (EA) is the complete, consistent and coherent set of methods, rules, models and tools which will guide the (re)design, migration, implementation and governance of business processes, organizational structures, information systems and the technical infrastructure of an organization according to a vision [6]. In this chapter we will discuss relevant Enterprise Architecture frameworks and their views on Requirements Engineering (RE).

3.1 TOGAF

In popular methods for enterprise architecture, such as The Open Group Architecture Framework (TOGAF, see figure 3)[17], (business) goals and requirements are central drivers for the architecture development process. In TOGAF's Architecture Development Method (ADM, see [17]), requirements management is a central process that applies to all phases of the ADM cycle. The ability to deal with changes in the requirements is crucial to the ADM process, since architecture by its very nature deals with uncertainty and change, bridging the divide between the aspirations of the stakeholders and what can be delivered as a practical solution.

TOGAF provides a limited set of guidelines for the elicitation, documentation and management of requirements, primarily by referring to external sources. TOGAF's content meta-model, part of the content framework, defines a number of concepts related to requirements and business motivation; however, this part has been worked out in little detail compared to other parts of the content meta-model, and the relation with other domains is weak. Also, the content framework does not propose a notation for the concepts. We do recognize the fact that requirements engineering drives architecture design. But TOGAF lacks the distinction between architecture as a design artifact and architecture as a frame of reference. In the former architecture is the result from RE, the latter uses architecture as a frame of reference to guide RE.

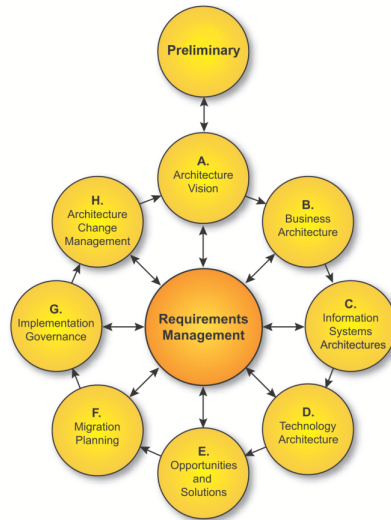


Fig. 3. TOGAF ADM

	WHAT	HOW	WHERE	WHO	WHEN	WHY	
	DATA	FUNCTION	NETWORK	PEOPLE	TIME	MOTIVATION	
SCOPE (contextual)	Cost of Things/Investment in the	Cost of Processes for	Cost of Locations in World	Cost of Organization	Cost of Timing/Cycle	Cost of Business	SCOPE (contextual)
Planner	Entity = Class of Business Thing	Process = Class of Business Process	Node = Major Business Location	People = Major Organizational Unit	Time = Major Business Event/Cycle	End-Motiv = Major Business Goal/Strategy	Planner
BUSINESS MODEL (conceptual)	e.g. Semantic Model	e.g. Business Process Model	e.g. Business/Logical Space	e.g. Work Flow Model	e.g. Business Schedule	e.g. Business Flow	BUSINESS MODEL (conceptual)
Owner	Entity = Business Entity Relationship = Business Relationship	Process = Business Process (P) = Business Business	Node = Business Location Link = Business Linkage	People = IT Organization Unit Block = Work Product	Time = Business Event Cycle = Business Cycle	End = Business Objective Means = Business Strategy	Owner
SYSTEM MODEL (logical)	e.g. Logical Data Model	e.g. Application Architecture	e.g. Distributed System Architecture	e.g. Access Interface Architecture	e.g. Processing Sequence	e.g. Business Rule Model	SYSTEM MODEL (logical)
Designer	Entity = Data Entity Relationship = Data Relationship	Process = IT Application Function (P) = Data Function	Node = IT System Location Link = Data Linkage	People = IT System Block = Software	Time = System Event Cycle = Processing Cycle	End = System Objective Means = System Strategy	Designer
TECHNOLOGY MODEL (physical)	e.g. Physical Data Model	e.g. System Design	e.g. Networking Architecture	e.g. Resource Architecture	e.g. Control Structure	e.g. Rule Design	TECHNOLOGY MODEL (physical)
Builder	Entity = Hardware/Software/Network/Storage/Device/Resource/Platform/Module	Process = Computer Function (P) = Data Function	Node = Hardware/Software Link = Data Linkage	People = User Block = Access Resource	Time = System Event Cycle = Computation Cycle	End = Capabilities Means = System Strategy	Builder
DETAILED REPRESENTATIONS (out-of-context)	e.g. Data Definition	e.g. Sequence	e.g. Network Architecture	e.g. Security Architecture	e.g. Timing Definition	e.g. Rule Specification	DETAILED REPRESENTATIONS (out-of-context)
Subcontractor	Entity = Field Relationship = Address	Process = Program Statement Code = Control Block	Node = Address Link = Physical Link	People = Identity Block = Job	Time = Interval Cycle = Reaction Cycle	End = Job condition Means = Step	Subcontractor
FUNCTIONING ENTERPRISE	e.g. DATA	e.g. FUNCTION	e.g. NETWORK	e.g. ORGANIZATION	e.g. SCHEDULE	e.g. STRATEGY	FUNCTIONING ENTERPRISE

Fig. 4. Zachman framework

In our view problem-oriented requirements engineering drives architectures design. This architecture design is then a solution to the experienced problems. This was already established by Michael Jackson and his problem frame approach [7] through a progression of problems.

3.2 Zachman Framework

The ancestor of Enterprise Architectures is the Zachman framework [26]. The framework as it applies to enterprises is simply a logical structure for classifying and organizing the descriptive representations of an enterprise that are significant to the

management of the enterprise as well as to the development of the Enterprise's systems. It was derived from analogous structures that are found in the older disciplines of Architecture/Construction and Engineering/ Manufacturing that classify and organize the design deliverables created during the process of designing or producing complex physical products.

Figure 4 presents and overview of the "Framework for Enterprise Architecture", usually known as the Zachman Framework. An important aspect in this framework is the motivation column. The motivation column explains why the architecture is needed. Looking at the motivation column we can already distinguish a link to requirements engineering. Explaining the motivation for either the architecture or the elements of the different architectural layers can be realized through problem-oriented RE.

3.3 Project Start Architecture

A technique to align architectures with solution realization is Project Start Architecture (PSA) [12] describes the relevant parts of the reference architecture at the start of a project. The PSA is a steering instrument that ensures the relevancy of the architecture in concrete projects. Architecture should not be an academic exercise by architects, but of concrete value in organizational change.

The PSA is a translation of general principles and guidelines relevant for the change projects. Relevant parts of the (reference) architecture are selected and written down in the PSA. The PSA is then handed down to the relevant project for solution realization. The solution realization process uses the PSA as an input document and validation document. The solution should use the boundaries set by the PSA. The PSA provides the context of solution realization; it does not describe the solution itself. The idea of PSA is very useful as it transfers scope and frame of reference to solution realization projects. It clearly defines the boundaries of the problem under investigation and solution designers can use this scope to specify detailed solution behavior.

3.4 I* for Enterprise Architecture Design

Eric Yu [14] [23] proposes to use I* as a problem investigation technique for architecture design and business modeling. This way the motivation for architectural elements is linked to their implementation. I* [24] is a technique that focuses on modelling and reasoning support for early phase requirements engineering. It tries to capture the understanding of the organizational context and rationales that lead up to systems requirements. It consists of two main modelling components. The Strategic Dependency (SD) model is used to describe the dependency relationships among various actors in an organizational context. The Strategic Rationale (SR) model is used to describe stakeholder interests and concerns, and how they might be addressed by various configurations of systems and environments [24].

I* can be used for both early and late phases of RE. During the early requirements phase *i** is used to model the environment of the system to be, it facilitates the analysis of the domain by allowing the modeller to diagrammatically represent the stakeholders of the system, their objectives and their relationships. During the late phases *i** models are used to propose the new system and the new processes and evaluate them on how well they meet the functional and non-functional needs of the users.

4 Alignment Architecture with Requirements Engineering

When analyzing the relevant literature from chapter 3 we can conclude the following on architecture-driven requirements engineering. Requirements play an important role in architecture design. We argue that the requirements for architecture design are problem oriented and the architecture provides the general design to these goals. This way architecture is seen as a design artifact, the solution for identified problems. Secondly we need to transfer the motivation and the architecture design into the solution realization projects. Here architecture is the frame of reference. Transferring the relevant parts of the architecture is the domain of Project Start Architecture (the solution itself or the solution blueprints are not in the domain of PSA). Although the documentation and theoretical integration into RE is weak (e.g. see [12]). To further investigate this we performed an exploratory case study, unfortunately we are unable to provide exact details of this case study due to confidentiality reasons. We can only provide some context in which the case study took place. To investigate our claims we elicited requirements from an off the shelf reference architecture and compared these to the results from traditional requirements elicitation techniques. This was done through eliciting requirements from the reference architecture based on the project goals. These requirements were compared to the results from the actual project. This way we were able to compare requirements elicited from architectural models with requirements from traditional techniques. We were able to show that we can use architecture to assist the elicitation and analysis of requirements, improve requirements specification and help validate the requirements.

4.1 Requirements Elicitation and Analysis

During requirements elicitation we were able to elicit a large number of requirements by inspecting the reference architecture. Furthermore, the traditional approach and the architecture-driven approach led similar requirements. The architecture-driven requirements were more general in nature. The reference architecture provided a scope for the problems and described possible solutions for these problems. We will elaborate this with an example. The company where the case study took place faced a problem about the integration of a new product in their current insurance portfolio. This new product required the use of an insurance broker. At the time of integrating this new product into the company they only sold insurances directly to their customers. This triggered a new problem since they had no idea how to implement and realize an insurance broker distribution channel and how to provide IT support for an insurance broker administration. Their architecture described this for them. For example, during a workshop a requirement emerged that for an insurance broker his name, address, bank account number and chamber of commerce data had to be recorded. When investigating the reference architecture we saw reference models describing recording insurance broker information and abstract examples of this information.

A second observation was that the architecture-driven approach facilitates the re-use of similar solutions as source for requirements elicitation. For example, when a new insurance product (or service) is introduced similar products could be used to

elicit change requirements. Lastly the architecture provided the relevant scope for the new systems through analyzing relationships between architectural elements.

One point of consideration is that the company used an off the shelf reference architecture. This architecture already described a desired to-be state, without a to-be analysis. For example, it provided solutions for problems that were not experienced yet. It therefore also provided ready to use solutions for problems. Secondly this way the organization had a very mature architecture to begin with, so it was quite easy to transfer the relevant models into solution realization projects.

4.2 Requirements Specification

During our case study we introduced a requirement specification template based on the architecture framework used in the organization. This template consists of business, application and technology layers (based on the meta-model used in their organization). This template was used to specify the requirements elicited from the architecture. The main argument to develop a template for a requirements specification around this meta-model is that solving an organizational problem is much more than just investigation and specifying the IT need. In section 2 we explained that our view includes a progression of problems. Using this template we were able to show which business problems were solved on the business layer and their relationship to the application level. Furthermore, we were able to show how specifying the requirements for a business service impacts and serves as input for specifying requirements for the supporting information systems. Thus, one could emphasize the underlying dependency relationships between the requirements positioned in the different layers of the above-mentioned template.

4.3 Requirements Validation

During requirements validation, the role of the enterprise architect is similar to that of any other stakeholder during the validation phase. In this setup the architect is regarded as stakeholder in the validation activities and may judge whether the specified requirements comply with the architecture goals, guidelines, principles, policies and constraints and with the architecture. Secondly, since the architecture described a desired to-be state it provided a validation mechanism in the form that a requirements specification should comply with.

5 Architecture-Driven Requirements Engineering

We have established that Requirements Engineering (RE) both happens to design architectures and realize the architecture. To design the architecture RE investigates the problematic phenomena, describes the business objectives and a way of working to realize these objectives. To realize the architecture we need to transfer the relevant requirements for the architecture and the architecture design into the solution realization projects. This way we heavily restrain the freedom of the solution designers to match the already established architecture.

5.1 Framework for Requirements Engineering

We argued that Requirements Engineering (RE) is about getting from problems to the possible solutions. Therefore we use a logical framework for problem solving [20] (see figure 5) as a RE framework.

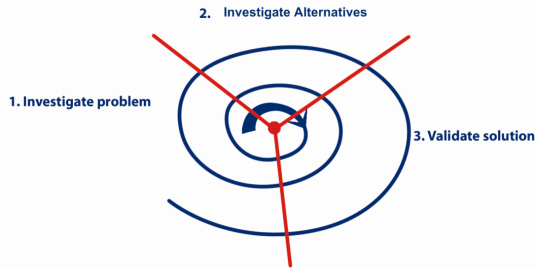


Fig. 5. Framework for requirements engineering

5.1.1 Problem Investigation

During problem investigation we take the problem-oriented view on requirements engineering. We find the stakeholders, record the relevant business objectives and specify how to reach these business objectives. This phase uses the concepts introduced in GORE (see KAOS [19] and i^* [14] [23]). This phase in our method leads to a goal tree that serves as input for the traditional requirements techniques. Important concepts during this phase are the stakeholders, their concerns, assessments of these concerns, goals (both hard and soft goals) and requirements. A precise definition and report on the design of the requirements language is out of the scope of this paper. But we will provide an exact syntax, to elaborate the example (see table 1).








5.1.2 Investigate Alternatives

In this step we start to look for possible solutions that are available to solve our problem. Solution specification is an important activity during this phase. Solution designers [5] propose system properties during this phase to reach the goals identified earlier. Solution alternatives range from proposing new (business) systems to actual alternative solution properties.

5.1.3 Solution Validation

In the solution validation phase the different solution alternatives are compared and analyzed. The main goal is to determine which solution best implements the business requirements [13]. Another important goal in this activity is to identify new problems. For example, when we have identified the need for a new service that we wish to provide to our customer and specified its desired behavior we are imposed with another problem. How are we going to realize this service internally? Other needed solutions might be adapted business processes, new information systems and a changed infrastructure. This leads to another cycle of the RE method.

Table 1. Elements from the requirements language

Abstract element	Concrete notation
Stakeholder	 Stakeholder
Concern	 Concern
Assessment	 Assessment
Hard goal	 Hard goal
Soft goal	 Soft goal
Requirement	 Requirement
Use case	 Use case

5.2 Framework for Architecture-Driven Requirements Engineering

In this framework (see figure 6) architecture is either a design artifact which requires requirements engineering or a frame of reference which guides requirements engineering.

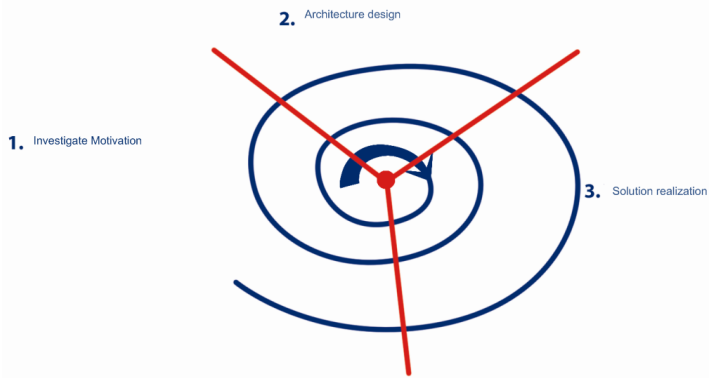


Fig. 6. Life cycle for architecture-driven RE

The framework from figure 5 applies to the framework from figure 6 as well. For example, we find the steps problem investigation, investigate alternatives and validate solution in the individual steps of this framework as well.

5.2.1 Investigate Motivation

Investigate problem

Before architecture design we investigate the motivation for the architecture. In requirements engineering terms, we investigate the business objectives and the way of working to reach these objectives. In the early stages, before architecture design we propose to use problem-oriented requirements engineering. More concretely, we have adopted a goal oriented approach. Goals are an excellent mechanism to explain the motivation for a solution [18]. If we compare this to existing GORE approaches, it resembles the early requirements phase found in i* [24].

Investigate alternatives

During investigate alternatives we propose the solutions for the particular problems depicted in the motivation plane. We also start specifying the solutions on a high level. For example initial use-case specification models. It is not required to provide detailed use-case specifications during this phase. When the solution designers start working on the solution they can take these use-case specifications as a starting point.

Validate solution

During solution validation the proposed solutions are evaluated and new problems are investigated. These relationships define the progression of problems defined by Jackson [7]. Solution validation during this phase focuses more matching the proposed solution to the goals and identifying new problems on an architectural level. For example in this setup, the stakeholder concerned with validation activities may judge whether the specified requirements comply with the enterprise architecture goals, guidelines, principles, policies and constraints [11].

5.2.2 Solution Realization

During solution realization we transfer the solutions from design plane and their motivation to the realization projects.

After the architecture is designed we need to transfer the motivation and the architectural models to the solution realization projects. A solution for this is found in Project Start Architecture (PSA) introduced by DYA [12]. The models defined here should lead to a blueprint of requirements that the requirements engineers can use for their solution specification.

Problem investigation

We now know what parts of the architecture are relevant and we might have solution blueprints. The architectural model here steers the requirements elicitation process. When we have exhausted this way of requirements elicitation, traditional techniques, like workshops and scenario elicitation can supplement our first draft of the requirements specification. The advantages of working this way is that the requirements elicitation activities get a head start and are constrained by the relevant parts of the organization, depicted in an architectural model. Architecture helps the requirements engineer with elaborating the relevant scope of the problem under

investigation. For example, when we know that the change goal for our project is to develop a new business service that allows customers to administer and maintain their insurance portfolio over the internet. The architecture can then provide the relevant models for insurance products, insurance selling processes, etc.

Investigate alternatives

During this phase we take a much more traditional approach. We investigate alternatives constrained to the solution we have to realize.

We use solution specification techniques for detailed solution specification. In terms of an IT system think of techniques from Object Oriented Analysis or Structured Analysis. These techniques are found in solution oriented requirements engineering. For business solutions, specification techniques from the business domain could be used. For example service blueprinting for a business service.

Solution validation

During solution validation we compare the different possible solutions to the system objectives. Validation is about to show which solution is expected to reduce the gap between the experienced problems and the desires.

6 Example

In this section we will provide an example case for our requirements engineering method. We will demonstrate how to use architectural models during problem investigation, solution specification and solution validation.

PRO-FIT is an average sized financial service provider, specialized in different insurance packages, such as life insurances, pensions, investments, travel insurances, damage insurances and mortgages.

In the last years PRO-FIT went through a structural change process, the result of which is that all business processes are consistent up the department level. However, the financial branch is one of the most dynamic and the senior management of PRO-FIT is now aware of new developments and threats, which require PRO-FIT to think of new ways to deal with these new challenges.

During the identification of new developments and threats the senior management of PRO-FIT became aware of the new service-oriented way of thinking. A market analysis identified a number of opportunities; one of them is a differentiation strategy for their insurance services using modern technology.

During the past few months the customer support at PRO-FIT identified a number of problems as well. Customers are complaining about the lack of insight in their insurance portfolios, competitors offer new internet based solutions where customers can request all kinds of information about their insurance portfolios.

During the remainder of this example we will use the requirements language depicted in table 1.

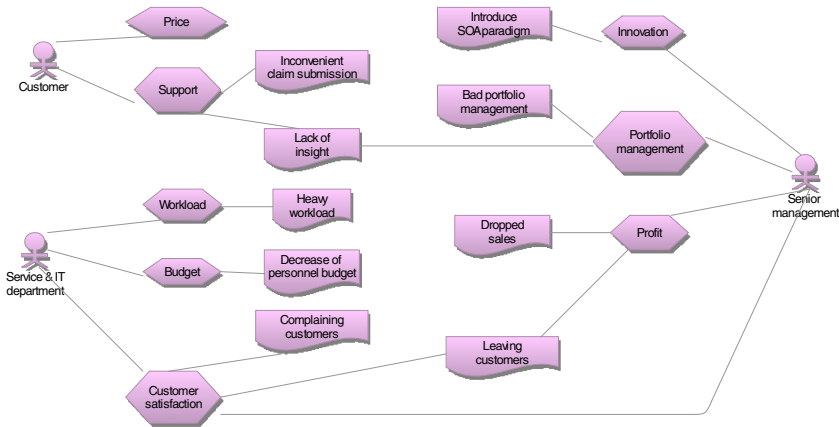


Fig. 7. The stakeholders, concerns and assessments

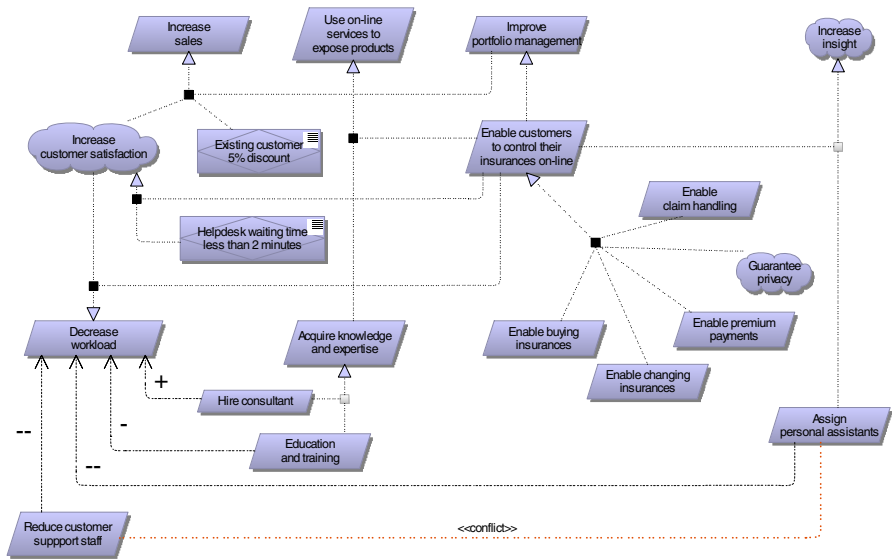


Fig. 8. Results from problem oriented requirements engineering

6.1 Investigate Motivation

During this step we will investigate the motivation of PRO-FIT. We will explore the stakeholders, their concerns and assessments. These concerns and assessments lead to goals. In this step architecture is a design artifact.

6.1.1 Investigate Problem

Because of space limitations we will restrict this example to three relevant stakeholders, with a limited number of concerns and assessments. We assume that we have a customer who is concerned with price and support. We also have a stakeholder (or stakeholders) senior management. Senior management is concerned with innovation, portfolio management and profit. Thirdly, we have identified the customer service department as a stakeholder. They are concerned with workload, budget and customer satisfaction. See fig. 6 for an overview of the concerns and assessments.

As we can see the identified concerns from the respective stakeholders can lead to assessments. These assessments are ways to address these concerns, for example the concern profit leads to an assessment of dropping sales. This is a threat to the organization and therefore needs to be addressed. This will lead to the high level goal “increase sales” (see figure 8). Through goal refinement we reach the goals that we want to introduce a new portfolio management service that allows the customer to buy insurances online, mutate his/her data online, pay their premiums and submit their claims.

6.1.2 Investigate Alternatives

During investigate solution alternatives we investigate the possible solutions which will realize our goals from section 5.1. In our case we will introduce a new portfolio management service. We use use-case specification to specify high level behavior. The use case portfolio management describes the high level behavior and can be refined into refined use-cases (see figure 9).

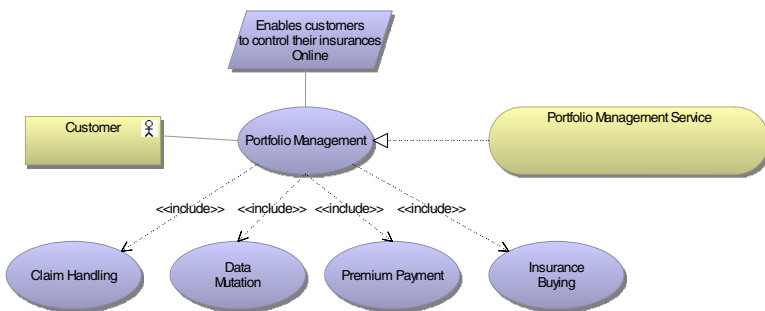


Fig. 9. New Portfolio Management Service

6.1.3 Validate Solution

During solution validation we both check the current specified solution and try to identify new problems. In this case it is determining the IT support. The solution defined in this chapter is then a problem for the IT specialist. In the next cycle of problem investigation and solution specification PRO-FIT assumes the role of a service consumer. During solution validation the architecture can be used to identify new problems based on the proposed solution. For example, during this example we introduced a new business service. This business service might introduce new business processes and it will need IT support. One way of finding new problems is to perform an impact analysis on the architecture [11].

6.2 Solution Realization

We argued that before solution realization starts, the architecture should be inspected for relevant information. This coincides with “architecture as a frame of reference”. In a best case scenario the architecture already describes a to-be state; this to-be state already provides a number of requirements and seriously limits the solution alternatives. When there is no to-be state, the architecture still provides relevant models, scope, context and structure to the RE activities. The relevant parts of the reference architecture comprise of guidelines, principles and the relevant models found. In this section we will provide a selection of the relevant models from the PRO-FIT architecture. We will realize a more business oriented solution, but this way of thinking also applies for IT based systems.

Because we know that we want to sell insurances we can select the product architecture for product information. We can also select the processes “claim handling” and “new insurance request”. Using the business information model we can already select the relevant information requirements. The reference architecture also describes PRO-FITS insurance portfolio, namely car insurances, life insurances, travel insurances and general liability insurances.

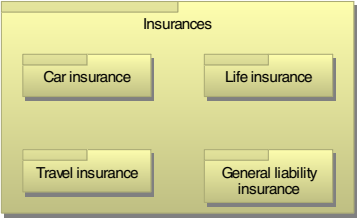


Fig. 10. The product architecture of PRO-FIT

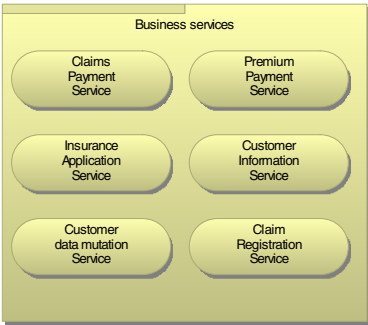


Fig. 11. The business service architecture of PRO-FIT

In figure 6 we illustrate the business services PRO-FIT delivers to its customers or internal departments.

Processes

In the BIP phase guidelines where given that the new service should support selling insurances, changing insurances, claim handling and premium payments. For illustration purposes we selected the relevant process models for closing contracts and claim handling. Closing contracts is the internal procedure for handling insurances requests.

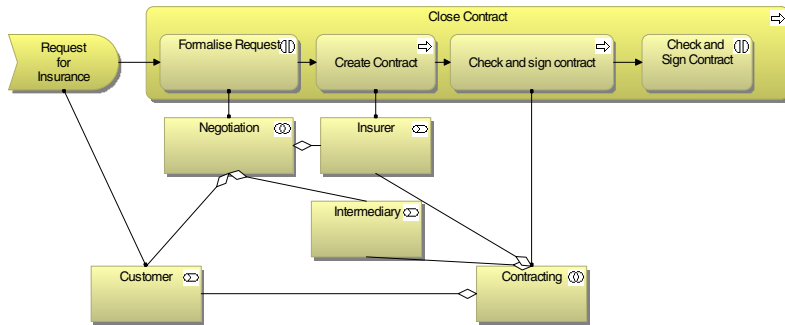


Fig. 12. Close Contract business process including the relevant entities and roles

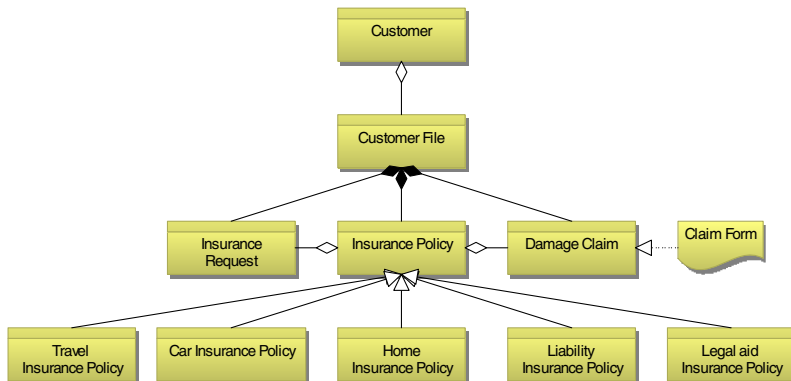


Fig. 13. Business information in PRO-FIT

Business Information

The architects also need to provide the relevant business information parts of the architecture. They already identified the relevant products and processes. In we depict PRO-FITS current business information model, as recorded in the reference architecture.

Another important aspect is to transfer the motivation for the architecture. The solution designers have to use the scope identified for the architecture. Secondly the use-cases specified at the architecture restrain the scope for the solution designers.

6.2.1 Problem Investigation

During problem investigation the architecture can provide the requirements engineer with the relevant models to determine the scope. After the business information

planning phase we know the main goals and that PRO-FIT wants to realize a new portfolio service. Through asking ourselves how and why questions we can refine the goal tree from figure 3. First sources of information are the architectural principles depicted in figure 4. The client satisfaction goal should be used for every solution realization project. The business function model also provides the relevant refinement goals (or relevant process models). An architecture driven way of working does not mean it replaces the traditional soft techniques like workshops and interviews. It is a supporting phase to get a head start. The results from architecture driven elicitation should be used as an input for the traditional techniques. It is even possible to refine the goal “support insurance selling” with “sell liability insurances”, “sell car insurances” using the product architecture.

During the solution realization we can also elicit requirements that realize “provide security”. Supporting “user identification” is a goal that refines “provide security”.

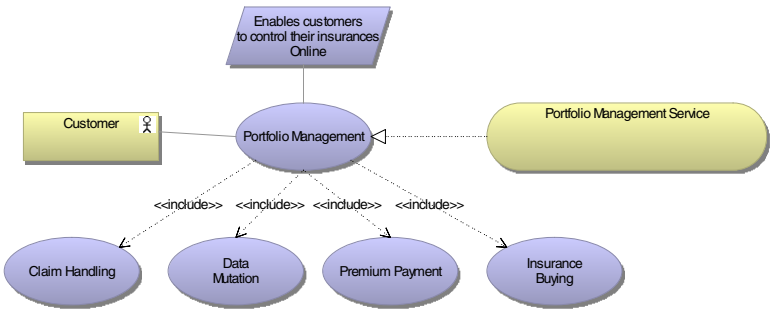


Fig. 14. Reuse of architecture solution specification

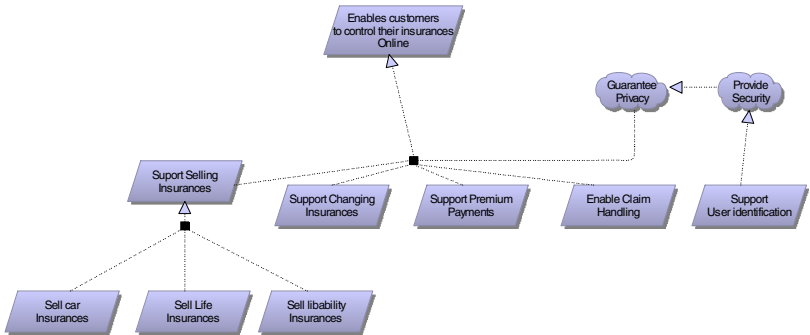


Fig. 15. Extension of the motivation

As mentioned before the exact details of these requirements are made clear using the traditional techniques. As we saw with our case study, the architecture provides less detailed requirements. Situational details should be elicited the old fashioned way. Another example is the relevant business information. Inspecting the architectural model from figure 13 provides the layout of the business information. Adding details to the objects is still required.

6.2.2 Investigate Solution Alternatives

In this step the person responsible for specifying the solution investigates the possible alternatives. For example in figure 16 he identifies two requirements that realize the goal “support user identification”. He has two possibilities here, using either Digi-ID or some form of biometric identification.

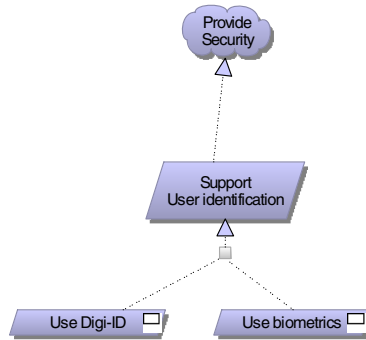


Fig. 16. Determining solution properties

In figure 17 we see the requirements “use i-deal” or “use credit cards” are possible alternatives to realize “support premium” payments.

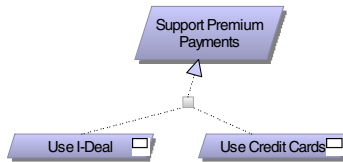


Fig. 17. Solution alternatives for support premium payments

A second step during this phase is specifying solution behavior. In figure 18 we demonstrate how the earlier use-cases from the previous phase are refined into more concrete solution behavior.

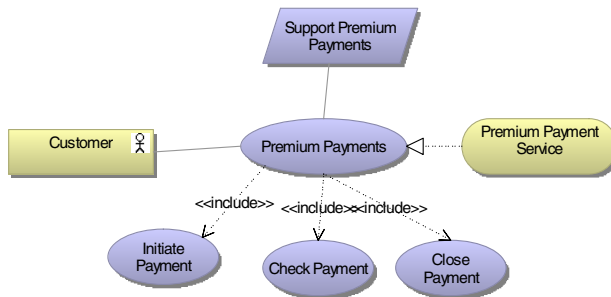


Fig. 18. Solution specification for premium payment Service

Solution specification does not end here. In figure 13 we provided the business information model for PRO-FIT. These initial data requirements can then be supplemented using the traditional techniques like workshops, interviews and scenario based elicitation.

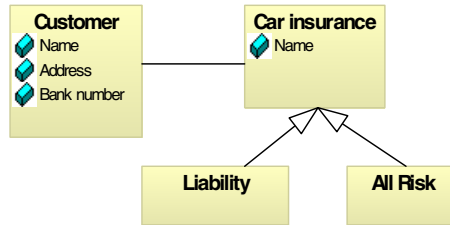


Fig. 19. Adding details to the object models from the architecture

6.3 Solution Validation

During solution validation we both check the current specified solution and try to identify new problems. In this case it is determining the IT support. The solution defined in this chapter is then a problem for the IT specialist. In the next cycle of problem investigation and solution specification PRO-FIT assumes the role of a service consumer. During solution validation the architecture can be used to identify new problems based on the proposed solution. For example, during this example we introduced a new business service. This business service might introduce new business processes and it will need IT support. One way of finding new problems is to perform an impact analysis on the architecture [11].

7 Concluding Remarks and Future Work

In this paper we have described the influence of the Enterprise Architecture (EA) paradigm on the way in which Requirements Engineering (RE) is performed. An extensive survey and classification of existing literature has shown that the link between these two areas is still weak. For a large part, the results described in this paper are based on the observations made during a practical case study carried out within a large Dutch insurance company.

In the first part of the paper, we have shown that a company's enterprise architecture can be a useful source for the elicitation of a large starting set of requirements. These may subsequently be refined using traditional requirements elicitation techniques, such as scenarios, workshops, interviews or surveys. This approach has a number of potential advantages: (1) time savings, among others because requirements may be reused between different projects; (2) the architecture places the requirements in their organizational context, which makes it easier to validate them with business stakeholders; (3) the architecture provides a way to structure requirements, which makes it easier to check for quality aspects such as consistency and completeness.

In the second part of the paper, we have made the combined approach to EA and RE operational by proposing a method for architecture-driven requirements engineering. This

method includes a process (way of working) and concepts for modeling requirements and their relationship to other concepts in the enterprise architecture. The method has been illustrated with a practical example.

As future work, we intend to fully validate our method in the pilot project we are currently carrying out. Although we have shown that it is possible to elicit requirements from enterprise architectures, we still do not know exactly how much improvement architecture-driven requirements engineering can actually offer. For example, how much of solution specification can we realize based on results from an architecture-driven elicitation process? How much faster is an architecture driven approach?

Secondly we need to extend the framework described here with analysis possibilities. For example, the stakeholder concerns are similar to the viewpoints in viewpoint oriented RE [16]. Identifying standard viewpoints or methods for viewpoint identification is a logical next step.

Another interesting topic for future research is the relationship between service-oriented computing and requirements engineering. The ideas from service orientation may further facilitate the reuse of requirements and solutions, thus speeding up the requirements engineering phase. However, service-oriented solutions may also lead to change in the requirements engineering process. In particular, we envisage that separate (complementary) requirements engineering processes are needed for the service provider and the service user.

Acknowledgements

This work wouldn't have been possible without the input from Dick Quartel, in the form of ARMOR, the requirements modeling language used in this paper.

References

- [1] Anton, A.I.: Goal-based requirements analysis. In: Proceedings of the Second International Conference on Requirements Engineering, pp. 136–144 (1996)
- [2] Aurum, A., Wohlin, C.: Engineering And Managing Software Requirements. Springer, Heidelberg (2005)
- [3] Bray, I.K.: An Introduction to Requirements Engineering. Addison-Wesley, Reading (2002)
- [4] Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems* 8(3), 203–236 (2004)
- [5] Cross, N.: Strategic knowledge exercised by outstanding designers. *Strategic knowledge and concept formation III*, pp. 17–30 (2001)
- [6] Iacob, M.E., Franken, H., Van den Berg, H.: Enterprise Architecture Handbook. Bizzdesign academy publishers (2007)
- [7] Jackson, M.: Software requirements & specifications: a lexicon of practice, principles and prejudices (1995)

- [8] Jacobson, I.: The use-case construct in object-oriented software engineering. *Scenario-based Design: Envisioning Work and Technology in System Development*, 309–338 (1995)
- [9] Jonkers, H., van Burren, R., Arbab, F., de Boer, F., Bonsangue, M., Bosma, H., ter Doest, H., Groenewegen, L., Scholten, J.G., Hoppenbrouwers, S., et al.: Towards a language for coherent enterprise architecture descriptions. In: *Proceedings of Seventh IEEE International Enterprise Distributed Object Computing Conference*, 2003, pp. 28–37 (2003)
- [10] Kotonya, G., Sommerville, I.: Requirements engineering with viewpoints. *Software Engineering Journal* 11(1), 5–18 (1996)
- [11] Lankhorst, M.: *Enterprise Architecture at Work: Modelling, Communication and Analysis*. Springer, Heidelberg (2005)
- [12] Luijckx, J.: White paper project start architectuur (in dutch). Sogeti internal report
- [13] Mulholland, A., Macaulay, A.L.: Architecture and the integrated architecture framework. Whitepaper, Capgemini, http://www.capgemini.com/services/soa/ent_architecture/iaf/
- [14] Samavi, R., Yu, E., Topaloglou, T.: Strategic reasoning about business models: a conceptual modeling approach. *Information Systems and E-Business Management*, pp. 1–28
- [15] Schoman, K., Ross, D.T.: Structured Analysis for requirements definition. *IEEE Trans. on Software Engineering* 3(1) (1977)
- [16] Sommerville, I., Sawyer, P., Viller, S.: Viewpoints for requirements elicitation: a practical approach. In: *Proc. Third IEEE International Conference on Requirements Engineering (ICRE 1998)* (1998)
- [17] The Standish The Open Group. *Togaf™ version 8.1.1 enterprise edition* (2006), <https://www.opengroup.org/architecture/togaf8-doc/arch/>
- [18] van Lamsweerde, A.: Goal-oriented requirements engineering: a roundtrip from research to practice. In: *Proceedings of 12th IEEE International Requirements Engineering Conference*, 2004, pp. 4–7 (2004)
- [19] van Lamsweerde, A., Letier, E.: From object orientation to goal orientation: A paradigm shift for requirements engineering. In: Wirsing, M., Knapp, A., Balsamo, S. (eds.) *RISSEF 2002. LNCS*, vol. 2941, pp. 325–340. Springer, Heidelberg (2004)
- [20] Wieringa, R.: *Requirements Engineering: Frameworks for Understanding*. Wiley, Chichester (1996)
- [21] Wieringa, R.: Requirements engineering: Problem analysis and solution specification (extended abstract). In: Koch, N., Fraternali, P., Wirsing, M. (eds.) *ICWE 2004. LNCS*, vol. 3140, pp. 13–16. Springer, Heidelberg (2004)
- [22] Wieringa, R., Heerkens, H.: Requirements engineering as problem analysis: Methodology and guidelines. Technical report, University of Twente (2003)
- [23] Yu, E., Strohmaier, M., Deng, X.: Exploring Intentional Modeling and Analysis for Enterprise Architecture (2006)
- [24] Yu, E.S.K., Mylopoulos, J.: Understanding why in software process modelling, analysis, and design, pp. 159–168 (1994)
- [25] Zachman, J.A.: Enterprise Architecture: The Issue of the Century. *Database Programming And Design* 10, 44–53 (1997)
- [26] Zachman, J.A.: A Framework for Information Systems Architecture. *IBM Systems Journal* 38(2/3), 454–470 (1999)