THE Open GROUP
Making standards work®

Supporting Requirements Management in TOGAF™ and ArchiMate®

A White Paper by:

Wilco Engelsman, Henk Jonkers, and Dick Quartel

February 2010

Supporting Requirements Management in TOGAF™ and ArchiMate®

Document No.: W101

Published by The Open Group, February 2010.

Any comments relating to the material contained in this document may be submitted to:

The Open Group
44 Montgomery St. #960
San Francisco, CA 94104

or by email to:

ogpubs@opengroup.org

Table of Contents

Boundaryless Information Flow™
achieved through global interoperability
in a secure, reliable, and timely manner

Executive Summary

This White Paper discusses methodology and language support for requirements management in TOGAF. The main contributions are:

- *A way of thinking*, which describes the philosophy, principles, and assumptions underlying the way requirements management is approached
- *A way of working*, which describes a reference method that identifies and structures the requirements management-related steps in the ADM phases and facilitates the re-use of existing requirements engineering techniques to support these steps
- *A way of modeling*, which describes concepts for modeling goals, requirements, stakeholders, their concerns, and assessments of these concerns, including the relationships between these concepts
- *A way of supporting*, which describes techniques for viewing and analyzing requirements models and tool support for creating and working with these models

Need for Support

Requirements management is an important activity in the process of designing and managing enterprise architectures. Requirements from various stakeholders form the basis for any change to an organization and its architecture. The quality of these requirements, the extent to which they are realized, and the ease with which they can be changed, determine the quality of any enterprise architecture.

Nonetheless, many enterprise architecture modeling techniques focus on *what* the enterprise should do by representing "as-is" and "to-be" architectures in terms of informational, behavioral, and structural models at the different architectural layers (e.g., business, application, and technical infrastructure). Little or no attention is paid to represent (explicitly) the reasons; i.e., the why, behind the to-be architectures in terms of motivations, rationale, goals, and requirements.

Inadequate requirements management is one of the main causes of impaired or failed IT projects [7], due to exceeding budgets or deadlines, or not delivering the expected results. As mentioned by Brooks: [2]: "No other part of the work so cripples the resulting system if done wrong". In this White Paper we mainly focus on requirements for enterprise architectures. Just like requirements found for IT solutions, the requirements for enterprise architectures are equally important.

Therefore, the requirements management process and the architecture development process need to be well-aligned, and traceability should be maintained between requirements and the architectural elements that realize these requirements. In the TOGAF Architecture Development Method (ADM) [10], requirements management is a central process that applies to all phases of the ADM cycle. However, TOGAF presents "requirements" on requirements management, and refrains from mandating or recommending existing methods and tools from the area of requirements engineering. Different methods and tools may fit different situations, and a choice among them should therefore be left open.

In this White Paper, we propose that the requirements management process in TOGAF be updated to provide more structure to facilitate its application in the ADM and the selection of existing methods and tools to support the process. Therefore, this document proposes a reference method consisting of:

- A generic "requirements engineering cycle" that can be repeated in each phase of the ADM. This cycle consists of three steps: problem investigation, proposition of (alternative) solution(s), and selection and validation of a solution.
- A "framework" that positions existing requirements engineering techniques in the aforementioned cycle. This framework facilitates the configuration of some requirements management process such that it fits the organization at hand by re-using existing requirements engineering techniques in the steps of the "requirements engineering cycle" and the ADM phases.

In addition, we propose that the ArchiMate language [6] should be updated to enable support for the modeling of requirements. ArchiMate complements TOGAF by defining a modeling language [8]. However, the current version of ArchiMate does not provide concepts for requirements modeling. Therefore, this document proposes such concepts, including their relationships to the existing ArchiMate concepts.

Finally, a method and language for requirements management should be accompanied by viewpoint and analysis techniques in order to facilitate the communication, documentation, and reasoning about requirements models and the requirements management process. Therefore, this document also discusses some basic requirements modeling viewpoints and techniques to analyze requirements models.

Requirements Chains – Way of Thinking

Requirements engineering (RE) is concerned with the process of finding a solution for some problem. This concern can be approached from a *problem-oriented view*, which focuses on understanding the actual problem, and a *solution-oriented view*, which focuses on the design and selection of solution alternatives. RE as problem analysis stems 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. A popular technique within this view is Goal-Oriented Requirements Engineering (GORE). GORE enables a number of analyses. Firstly, it facilitates reasoning about the motivation and justification of a proposed solution. Goal models can be analyzed to demonstrate which goals realize or motivate other goals and which goals conflict or negatively contribute to other goals. Secondly, GORE demonstrates the contribution of the proposed solution to the actual need.

The second view on RE stems from software engineering. In this view RE is seen as solution specification. Using this view the requirements engineer specifies the context in which the system will operate, provides a list of desired system functions, defines the semantics of these functions, and provides a list of quality attributes of those functions. This is the more traditional view on RE. Techniques and methods in this view are based on Structured Analysis (SA) and Object-Oriented Analysis (OOA). Structured analysis focuses on the flow of data and control of the system to-be. Object-oriented analysis applies object modeling techniques to analyze the functional requirements of the system to-be. An important OOA technique is use-case elicitation and specification. Use-cases capture the solution behavior in terms of interaction scenarios between the system and its user.

When we look at RE from an engineering perspective we can identify so-called *problem chains*, where each chain links a problem to a solution such that the solution is considered again as a problem by the next chain. For example, a business analyst may investigate a business problem and specify a business solution for this problem. This new solution most likely needs IT support, therefore becoming a problem for the IT analyst.

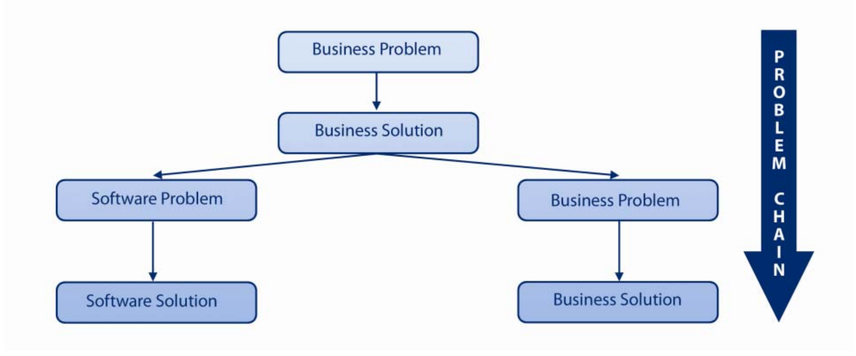

Figure 1: Problem Chains

Problem chains link requirements engineering to enterprise architecture. This is illustrated in Figure 2. The *why* column represents the problem-oriented view and defines the business needs, goals, requirements, and use-cases that should be addressed. The *what* column represents the solution-oriented view in terms of enterprise architecture elements such as services, processes, and applications. These architecture elements define *what* the enterprise must do to address the business needs, goals, requirements, and use-cases. At the same time, these requirements engineering elements motivate and justify *why* the enterprise architecture is defined the way it is.

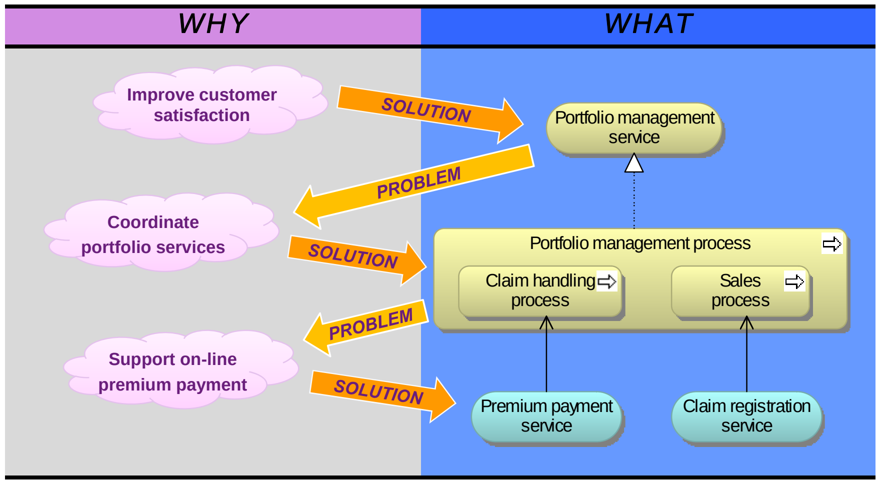
WHY
WHAT
Improve customer satisfaction
SOLUTION
Portfolio management service
PROBLEM
Coordinate portfolio services
SOLUTION
Portfolio management process
Claim handling process
Sales process
PROBLEM
Support on-line premium payment
SOLUTION
Premium payment service
Claim registration service

Figure 2: Linking Requirements to Architecture, and Vice Versa

For example, an organization may experience dropping sales, unsatisfied customers, and a high workload for the customer support department. In order to address these problems, the organization decides to introduce a new business service to support online portfolio management. This solution leads to a new design problem: the creation of new or the adaptation of existing business processes that support this service. Similarly, these processes may require IT support, which leads to the development of new application services.

This way of thinking enables traceability. Enterprise architecture elements can be traced back to the goals and requirements that motivated their introduction. Conversely, requirements engineering elements can be traced forward to the services, processes, and applications that implement these elements. This traceability is needed to successfully analyze and manage the impact of changes to an enterprise. For example, in case certain business goals are affected by changes in legislation it now becomes possible to determine precisely which products and services are affected by these changes.

Architecting Requirements – Way of Working

Following the idea of problem chains enables an iterative way of working. Given some problem, the first step is to analyze the problem and elicit goals and requirements that address the problem. These goals and requirements are represented by a requirements model. The second step is to conceive a composition of products, services, processes, and applications that realizes the goals and requirements. This composition is represented by an architecture model. Both steps can again be repeated for (the problem of) realizing the elements of the architecture.

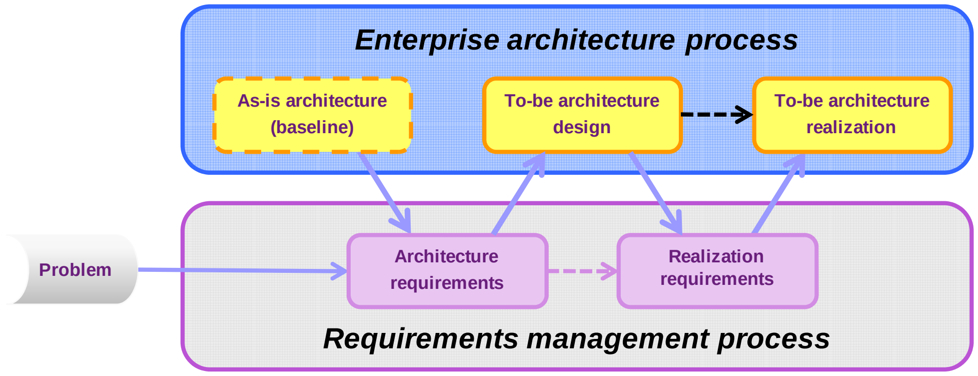
Enterprise architecture process
As-is architecture (baseline)
To-be architecture design
To-be architecture realization
Problem
Architecture requirements
Realization requirements
Requirements management process

Figure 3: Process for Architecture-Driven Requirements Engineering

Figure 3 illustrates the relationship between requirements models and architecture models and, indirectly, also the relationship between the requirements management and enterprise architecture processes. We consider *requirements management* as the combination of requirements engineering; i.e., identifying, analyzing, refining, and modeling requirements, and requirements maintenance; i.e., communicating and tracing requirements. Enterprise architecture represents the process of translating these requirements into architecture models. These processes are typically divided into distinct phases, which results in a series of requirements and architecture models such that models in succeeding phases refine models from preceding phases (as represented by the dashed arrows). For example, Figure 3 illustrates a process of two phases: the design and realization of some enterprise architecture.

Requirements Engineering Cycle

The idea of problem chains distinguishes two views on an architecture model: first as a design artifact that represents a solution for some design problem, and second as a frame of reference that delimits the design or solution space. These views are illustrated in the left part of Figure 4.

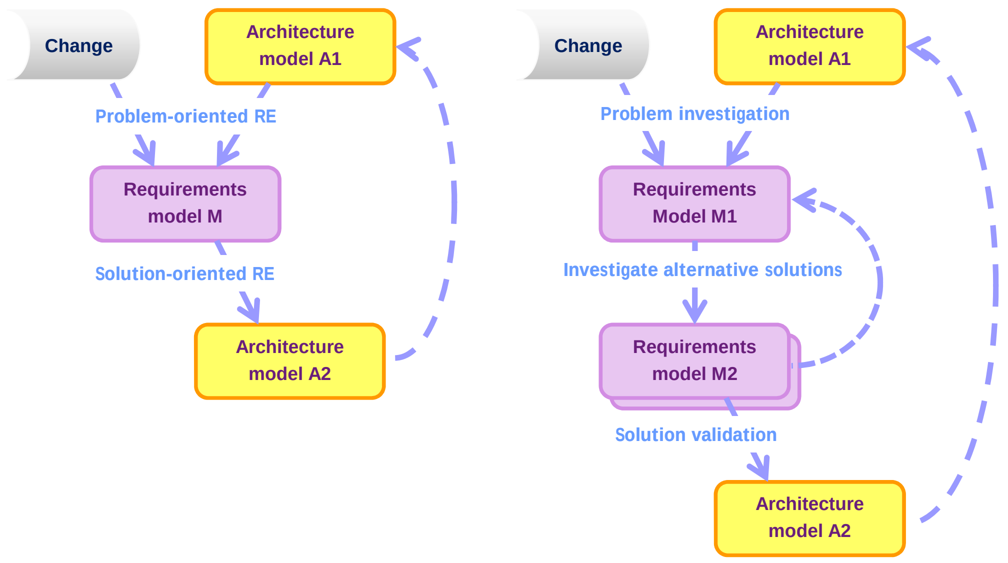

Figure 4: Architecture as Design and as a Frame of Reference

In general, requirements engineering starts with some organizational change that needs to be addressed. This problem cannot be approached "from scratch", but has to take the current organization into account, as represented by architecture model A1. This means that any requirement or goal in requirements model M should be defined "relative" to architecture A1 in order to address the change. In this situation, architecture A1 acts as a frame of reference for (problem-oriented) requirements engineering. Subsequently, a new architecture A2 is designed that realizes a solution for the requirements and goals in model M. In this situation, architecture A2 is considered a design artifact that results from (solution-oriented) requirements engineering.

The right part of Figure 4 depicts a further decomposition of requirements engineering into three steps:

- **Problem investigation**, which focuses on the problem – i.e., the organizational change – by identifying and analyzing its cause in terms of the involved stakeholders and their concerns, and by setting goals to deal with the change.
- **Investigate solution alternatives**, which refines the goals in order to find possible solutions to realize them. Analyses are performed to reveal conflicts between (refined) goals or contribution relations between goals in which goals contribute positively or negatively to other goals. Typically, these analyses trigger the identification and elaboration of alternative solutions.
- **Solution validation**, which validates alternative solutions and chooses the "best" among them. This choice is influenced, amongst other things, by the conflict and contribution relations that have been identified.

These steps constitute a generic requirements engineering cycle (RE cycle) that can be repeated at successive phases in the development of some enterprise architecture, as indicated by the dashed arrows in Figure 4. Furthermore, the identification, analysis, and refinement of solution alternatives in the second step may be repeated as well, leading to "sub-cycles".

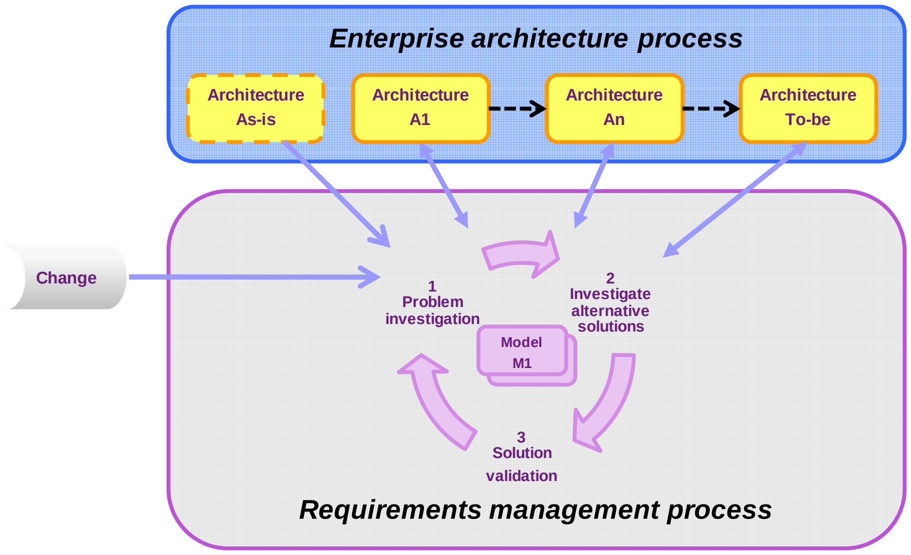

Figure 5: Application of the Requirements Cycle

Figure 5 illustrates the repeated application of the RE cycle in the phases of some architecture development process. The models in the center of the cycle illustrate the requirements models that are produced during the application of the engineering cycle. It is important to observe that after the elicitation of some goal in a certain cycle, this goal not necessarily has to be refined towards a solution in the same cycle. Depending on the type of goal, a goal may have to wait until the proper design phase before it is addressed. For example, goals that are related to technology may have to wait until later phases in which one decides about how enterprise applications are implemented using software and hardware.

Figure 6 depicts an alternative illustration of the application of the RE cycle, assuming the architecture development process is structured according to the TOGAF ADM. The requirements engineering cycle is not intended as a replacement of the Requirements Management phase in TOGAF; instead, it provides concrete guidelines to support requirements management.

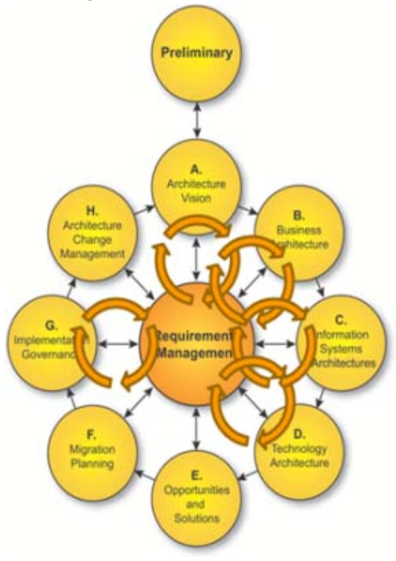
Preliminary
A. Architecture Vision
B. Business Architecture
C. Information Systems Architectures
D. Technology Architecture
E. Opportunities and Solutions
F. Migration Planning
G. Implementation Governance
H. Architecture Change Management
Requirements Management

Figure 6: Requirements Cycle in the TOGAF ADM

The requirements engineering cycle is applicable in most phases of the ADM.

Architecture Vision

The main requirements-related objectives of this phase are to validate the business goals, to define the relevant stakeholders, their concerns, and the goals and requirements that address these concerns, and to develop an architecture vision that presents a first solution. This vision consists of a high-level view of the baseline and target architecture. To validate the vision, TOGAF recommends a trade-off analysis and gap analysis. These activities fit very well in the presented reference method for requirements management. During the problem investigation step the business goals are investigated, stakeholders concerns are identified, and goals elicited to address these concerns. Based upon these goals, requirements are identified and assigned to the first high-level architectural vision elements during the investigate solution alternatives step. During the solution validation step we can perform a trade-off analysis and gap analysis.

Business, Information Systems, and Technology Architecture

Because the business, information systems, and technology architecture phases consist of similar activities, we discuss these phases in a single paragraph. The objective of these ADM phases is to elaborate the high-level baseline and target architecture at business, information systems, and technology level. This involves, among other things, a more detailed analysis of stakeholders and their concerns, and the refinement of business goals and requirements to address these concerns. One may choose to distinguish between different types of requirements, such as functional, non-functional, assumptions, and constraints. For validation purposes, TOGAF also recommends in these phases a trade-off analysis and a gap analysis. These activities again fit perfectly in the presented reference method. During problem investigation the previously identified solution is investigated. Additional stakeholders are identified and concerns elicited. Additional requirements (both functional and non-functional) are identified in the investigate solution alternatives step. During solution validation we can choose between the alternatives and select the best one. The business, information systems, and technology architectures are refined by the reference method in an iterative way. The results from the business architecture phase are used as a starting point for the problem investigation step of the information systems architecture phase, and the results of the information systems architecture phase are used again as a starting point for the problem investigation step of the technology architecture phase.

Opportunities and Solutions/Migration Planning

These two phases are concerned with setting-up the solution realization projects for realizing the architecture. Therefore, these steps do not contain any actual requirements engineering activities. Opportunities and solutions is the first phase which is directly concerned with the implementation of the target architecture. It identifies the programs, projects, and portfolios that deliver the target architecture from the previous phases. It uses the previously identified functional requirements to identify work products and realization projects. For these realization projects an implementation approach needs to be chosen; for example, COTS selection, bespoke systems specification, etc. This does effect the requirements engineering process that is responsible for solution realization, but falls outside the scope of TOGAF. Migration planning is concerned with how to move from the baseline architecture to the target architecture. The previously identified projects are prioritized and budgets assigned. The results from opportunities and solutions might result in changes to the architectural documents and corresponding requirements. We distinguish between architectural requirements and realization requirements. Realization requirements are the requirements identified in solution realization projects. They refine the architectural requirements, but new requirements may be identified as well depending on the chosen solution type.

Implementation Governance/Architecture Change Management

During these phases the actual solutions are realized and monitored to ensure compliance with the architecture. The previously identified projects are fed with the architectural requirements defined so far. Basically, this jump-starts the classical requirements engineering process. Classical requirements engineering is concerned with specifying the requirements for a solution (e.g., new applications). Requirements management in TOGAF is concerned with validating these detailed requirements and solution specifications to the architectural context and requirements. This fits the presented reference method. The architectural requirements form the starting point of the problem investigation step, in which additional stakeholders may be identified including concerns and goals related to the implementation of the architecture. Next, detailed requirements are elicited to investigate alternative implementation solutions. And finally, these solutions are validated against the architectural requirements, and a choice is made among the alternatives.

Positioning and Re-Use of Existing RE Techniques

Traditionally, the following steps are distinguished in requirements engineering:

- **Elicitation** is responsible for the *identification* of stakeholders, their concerns, the assessment of these concerns, and the goals that address the assessments. Existing techniques like workshops, surveys, and interviews can be used to perform and guide the elicitation process [5] [4] [11]. This results in a collection of stakeholders, concerns, assessments, and high-level goals. This collection is reflected in an initial requirements model, but typically without bothering too much (yet) about the relationships among the model elements.

- **Analysis** is responsible for the *structuring* and the *refinement* of the outcome of the elicitation step [1] [3]. Structuring is needed to check and improve the consistency and completeness of the model elements. For example, equivalent elements may be united, and conflicts or mutual dependencies among goals (e.g., two goals are both considered a sub-goal of each other) should be detected and resolved. Refinement is needed to investigate alternative solutions to realize the identified goals. Business scenarios and use-cases are popular techniques to support refinement. Finally, the completeness of the resulting requirements model is checked and priorities may be assigned to individual goals and requirements. A requirements model is complete if all goals and requirements are (backwards) *traceable* to their stakeholder and concerns, and *vice versa*; i.e., each stakeholder concern is (forwards) traceable to one or more goals that address this concern, and these goals are again (forwards) traceable to requirements or architectural elements, such as services and processes, that represent a solution to the goal.

- **Specification** is responsible for the *representation* of the models resulting from the elicitation and analysis steps, using some modeling or specification language. Most likely, this language is used already during the previous steps, such that this step may suffice by checking the syntax and semantics of the models.

- **Validation** is responsible for the *assessment* of the quality of the alternative solutions found in the analysis step, and for choosing the "best" alternative. The quality of a solution can be defined by its contribution to selected (soft) goals. The elicitation and analysis steps may have to be revisited to assess and model these contributions (see also Requirements Concepts – Way of Modeling). In addition, interviews and workshops may be used again to assess whether a solution meets stakeholders' expectations.

There is a clear correspondence between these steps and the steps of the requirements engineering cycle. The "problem investigation" step corresponds to the elicitation step and the structuring part of the analysis step.

The "investigate alternative solutions" step corresponds to the refinement part of the analysis step. And the "solution validation" step corresponds to the validation step. Specification is not considered a separate step in the requirements engineering cycle, because requirements models are created and manipulated during all steps of the cycle.

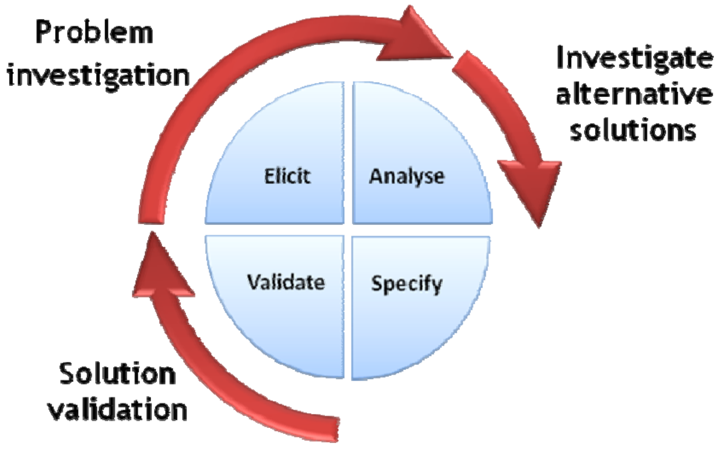

Figure 7: Framework Describing the Correspondence between Traditional RE Steps and the Steps of the RE Cycle

Figure 7 depicts the correspondence, which provides a "framework" for positioning and re-using existing requirements engineering techniques in the RE cycle of Figure 5. Despite this correspondence, we favor the steps of the RE cycle. The first and second steps distinguish clearly between the problem-oriented and solution-oriented character of requirements engineering. We consider this distinction essential when addressing problems and developing solutions for these problems.

Requirements Concepts – Way of Modeling

The current version of ArchiMate supports the modeling of *extensional* and *intentional* properties of an enterprise, in terms of informational, behavioral, and structural architecture elements. Extensional properties model – e.g., the products and services that are offered – and intentional properties model how they are offered by processes and applications.

Figure 8: Extension of the ArchiMate

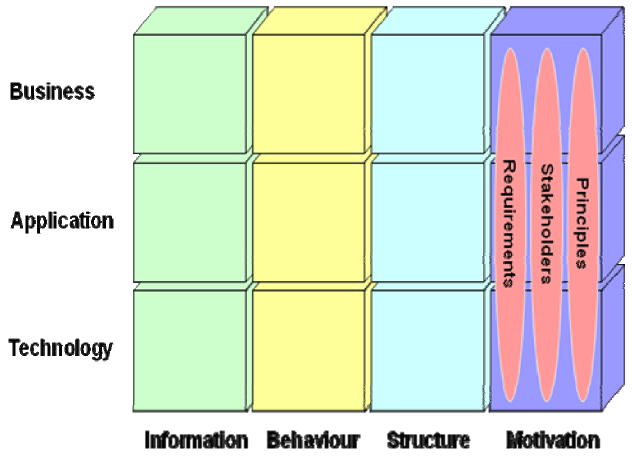

Figure 8: Extending ArchiMate

To support the modeling of intentional properties, the modeling framework underlying ArchiMate is extended with the motivation aspect. Figure 8 depicts this extension. The motivation aspect is concerned with the goals and intentions of the enterprise, and resembles the motivation (or why) column of the Zachman framework [12] [13].

We distinguish the following modeling domains across the motivation aspect: the stakeholder domain, the principles domain, and the requirements domain. The choice of modeling concepts within these domains has been motivated in [9].

Stakeholder Domain

This domain models the stakeholders of the enterprise, including their concerns and the assessment of these concerns. A concern is interpreted as some area of attention or interest. For example, a CEO may be concerned with executing the mission of the enterprise, a CIO with the clarity of the enterprise architecture and its ability to adapt to change, and a systems manager with the capacity and reliability of the computing and networking platforms used within the enterprise. These concerns may be assessed using a SWOT (Strengths, Weaknesses, Opportunities, and Threats) analysis. For example, this analysis may reveal that the enterprise's architecture lacks traceability, which makes it difficult to handle change.

In addition, the users or customers of the enterprise may be considered as stakeholders. Customers may be concerned with, e.g., the diversity of the products and services that are offered or the privacy of their information. Also these concerns may be assessed (not necessarily in terms of SWOT) to reveal customer needs. Stakeholders and their concerns may be identified from the enterprise's business plan.

The following table depicts the concepts of the stakeholders domain, including their notation.

Concept	Notation	Icon
Stakeholder	Stakeholder 👤	👤
Concern	Concern 👁	👁
Assessment	Assessment 🔍	🔍

Stakeholder – A stakeholder represents an individual, team, or organization (or classes thereof) with interests in, or concerns relative to, the outcome of the architecture. This definition is adopted from TOGAF [10]. Examples of stakeholders are the board of directors, shareholders, customers, business and application architects, but also legislative authorities.

Concern – A concern represents some key interest that is crucially important to certain stakeholders in a system, and determines the acceptability of the system. A concern may pertain to any aspect of the system's functioning, development, or operation, including considerations such as performance, reliability, security, distribution, and evolvability. This definition is adopted from TOGAF [10].

Assessment – An assessment represents the outcome of the analysis of some concern. This outcome may trigger a change to the enterprise architecture, which is addressed by the definition of new or adapted business goals.

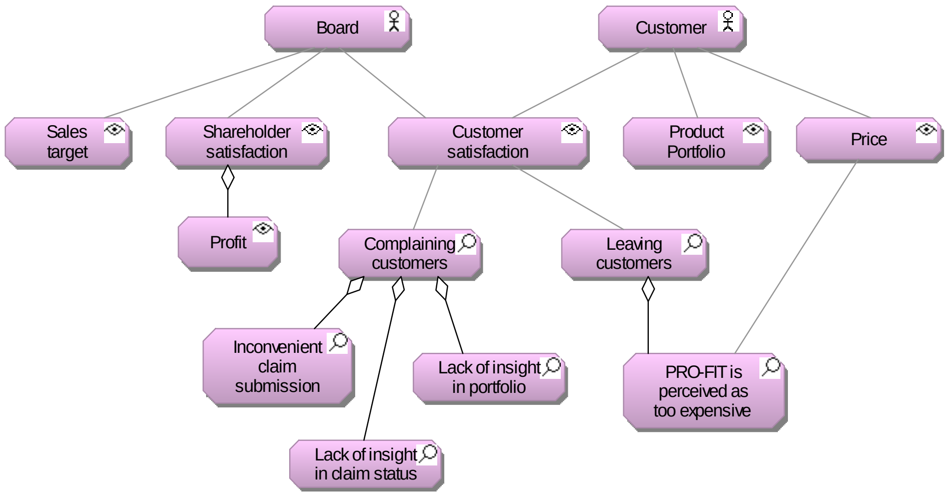

Figure 9: Stakeholder View of PRO-FIT

Relations – The association relation of ArchiMate is (re-)used to relate stakeholders to concerns and concerns to assessments. A stakeholder can have one or more concerns, and a concern may be shared by multiple stakeholders. An assessment typically assesses a single concern, but could involve multiple concerns. A concern may be analyzed through different assessments.

Figure 9 depicts part of the stakeholders' view of PRO-FIT, a company that sells insurances. This view shows two stakeholders: the "board" of PRO-FIT and the "customer" of PRO-FIT. Each stakeholder has a number of concerns, which may be shared, such as the "customer satisfaction" concern. This concern has been analyzed, leading to the assessments "complaining customers" and "leaving customers". In this case, the assessments are decomposed into sub-assessments to describe, e.g., the type of complaints in more detail.

Also stakeholders and concerns can be decomposed. For example, "profit" is modeled as a sub-concern of "stakeholder satisfaction".

Principles Domain

This domain models amongst other things the vision, mission, strategies, policies, principles, and guidelines of the enterprise, constituting the high-level constraints for the design of the enterprise architecture. This domain is not further elaborated in this White Paper.

Requirements Domain

This domain models the goals, requirements, and expectations that further constrain the design of the enterprise architecture. These goals, requirements, and expectations may originate from the constraints set in the principles domain or from the assessment of concerns in the stakeholders domain. This assessment may reveal strengths, weaknesses, opportunities, or threats that need to be addressed by means of changing existing goals or setting new ones.

The following table depicts the concepts and relations of the requirements domain, including their notation.

Concept	Notation	Icon
Hard goal	Hard goal	◎
Soft goal	Soft goal	⊛
Requirement	Requirement	▱
Use case	Use case	

Relation	Notation
Decomposition	◇————
Means-end	◁- - - - - -
Contribution	<- - +/- - -
Conflict	- - - ⚡- - -
Include	<- - <<include>>
Extend	<- - <<extend>>

Goal – A goal represents some end that a stakeholder wants to achieve. In principle, an "end" can represent anything a stakeholder may desire, such as a state of affairs, a produced value, a realized effect, or a created property. Examples of goals are: to increase customer satisfaction (to 85%), to support online portfolio management, to introduce a web portal, and to prefer open-source technology. To distinguish between precise and measurable goals and goals that are more "soft", so-called hard and soft goals can be defined, respectively. In addition, a goal may have attributes to define its name, specification, type, priority, assumptions, success criteria, and norms for these criteria.

A goal abstracts from how the "end" is realized and by whom. In order to model the realization of a goal, two types of refinement relations are distinguished:

- A **decomposition relation** relates a goal to multiple sub-goals, such that each sub-goal must be achieved to achieve the (super) goal. The decomposition relation is typically used to split a single complex goal into multiple smaller, more simple and detailed goals. The union of the sub-goals is sufficient to achieve the super goal. Decomposition of a goal into a single sub-goal is not recommended. In this case the goal and the sub-goal are considered as equivalent.

- A **means-end relation** relates an (abstract) goal to either a more concrete goal or an ArchiMate behavior element to a goal, such that the concrete goal or the behavior element realizes the (abstract) goal. In terms of its notation, achieving or realizing the means at the tail of the relation is necessary and sufficient to achieve the goal at the head of the relation. In order to represent alternative means for realizing a goal, a separate instance of the means-end relation has to be used for each of the means.

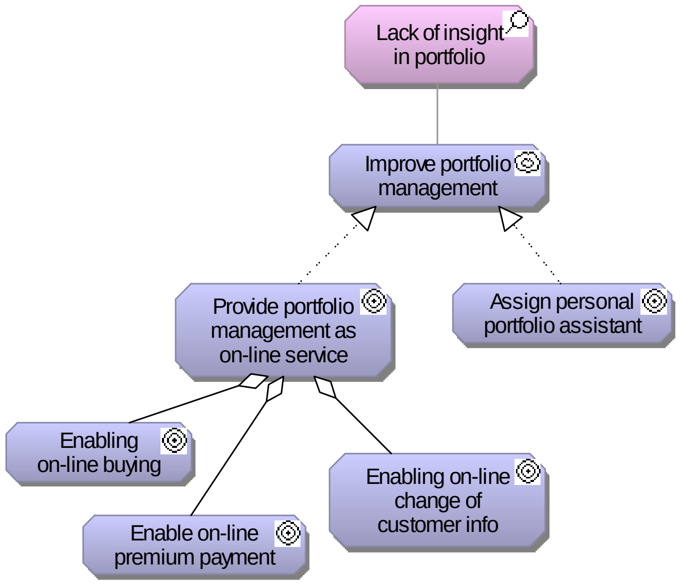
Lack of insight in portfolio
Improve portfolio management
Provide portfolio management as on-line service
Assign personal portfolio assistant
Enabling on-line buying
Enabling on-line change of customer info
Enable on-line premium payment

Figure 10: Goal Refinement Relations

Figure 10 depicts an example of both types of refinement relations. The assessment "lack of insight in portfolio" is addressed by the soft goal "improve portfolio management". Two hard goals – "provide portfolio management as online service" and "assign personal portfolio assistant" – are defined as alternative means to realize the soft goal. The former hard goal has been further decomposed into three sub-goals.

The modeling of alternative means to realize a goal implies that a selection has to be made among these means. In order to facilitate this selection, the **contribution relation** allows a modeler to represent that some (contributing) goal influences the achievement of another (contributed) goal positively or negatively. Depending on the level a detail, a range of values may be used; e.g., [0..10] or {++, +, - --}.

A positive contribution relation does not define that the contributing goal must be achieved to achieve the contributed goal; i.e., the achievement of the contributing goal is not a necessary condition (nor a sufficient condition). In contrast, the achievement of a sub-goal, as defined by the decomposition relation, is a necessary condition for the achievement of the super goal.

Similarly, a negative contribution relation does not define that the achievement of the contributing goal is a sufficient condition to prevent the achievement of the contributed goal. Instead, the **conflict relation** is used to relate two goals such that the achievement of one goal prevents the achievement of the other goal, and *vice versa*.

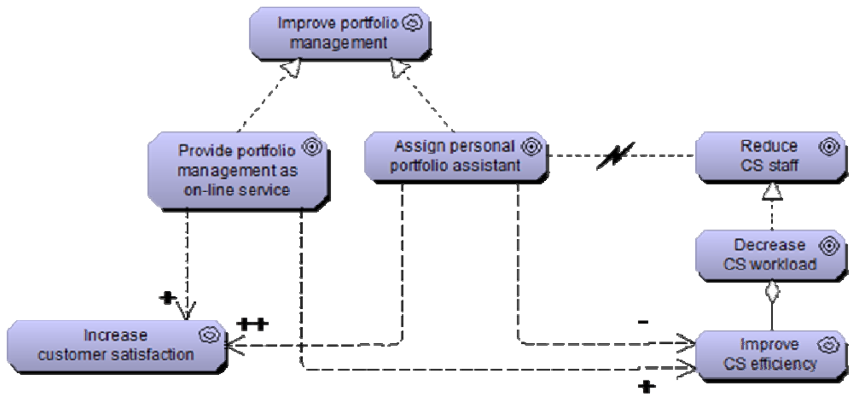
Improve portfolio management
Provide portfolio management as on-line service
Assign personal portfolio assistant
Reduce CS staff
Decrease CS workload
Increase customer satisfaction
Improve CS efficiency
+
++
-
+

Figure 11: Contribution and Conflict Relation

Figure 11 depicts an example use of the contribution and conflict relation. Means "provide portfolio management as online service" contributes positively to the soft goal "increase customer satisfaction", whereas the means "assign personal portfolio assistant" contributes even more positively. Based on these contributions, one might decide to choose the latter means as the best realization of the end "improve portfolio management". However, when considering the contribution to the soft goal "improve CS (customer support) efficiency", the former means contributes positively again, but the contribution of the latter means is negative. Furthermore, the latter means conflicts with a higher-level goal from which the soft goal "improve CS efficiency" was derived.

Requirement – A requirement represents some end that must be realized by a single actor. Similar to a goal, an end represents something that is desired, such as a state, value, effect, or property. Examples of requirements are the calculation of insurance premiums by an application component or the coordination of portfolio services by some business process.

The requirement concept is a specialization of the goal concept, since both concepts define some end, but the requirement concept also defines who or what is responsible for realizing the end. Furthermore, a requirement is assigned to a single actor. This implies that a goal may have to be decomposed into sub-goals, before an assignment to actors is possible.

Use-case – A use-case represents the interaction between a system and some external actor; e.g., a user. This actor typically initiates the use-case to achieve some goal. A use-case may describe alternative sequences (called scenarios) of interactions that fulfill the goal. In addition, a use-case may describe alternative scenarios that lead to failure. The "standard" UML notation for use-cases is used, and the common use-case relations "include", "extend", and "specialization" are supported.

A use-case is considered to be a kind of requirement, which defines from a user's perspective the functionality that is expected from the system. Two types of use-case can be distinguished:

- A **business use-case** is a use-case that represents the interaction between a business service provider and a business service consumer.
- A **system use-case** is a use-case that represents the interaction between an application service provider (application component) and an application service user.

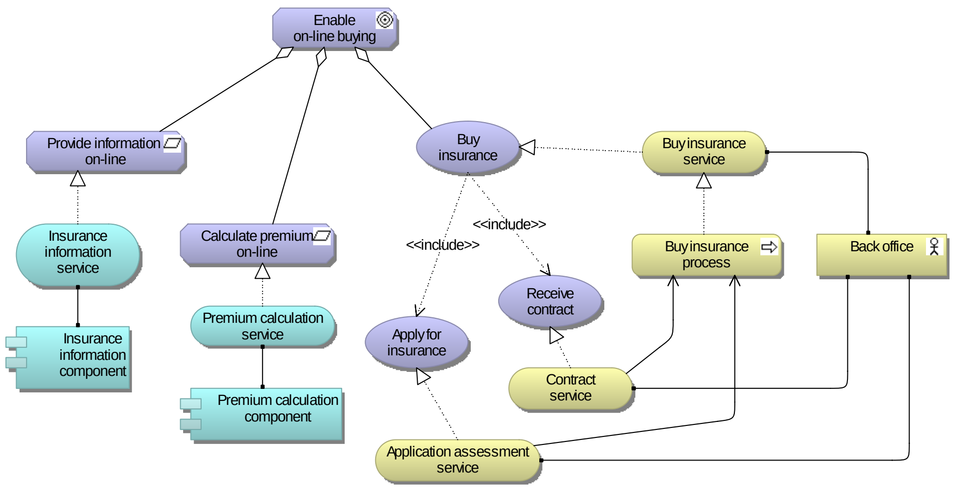
Enable on-line buying
Provide information on-line
Calculate premium on-line
Buy insurance
Buy insurance service
Insurance information service
Insurance information component
Premium calculation service
Premium calculation component
<<include>>
<<include>>
Receive contract
Apply for insurance
Buy insurance process
Back office
Contract service
Application assessment service

Figure 12: Requirements and Use-Cases

Figure 12 depicts the decomposition of the goal “enable online buying” into the requirements to provide online information about insurances, to calculate the premium of selected insurances, and to support the actual buying of some insurance. The latter requirement is modeled as a use-case, which includes two smaller use-cases; i.e., the application for some insurance and the receipt of the insurance contract. Each requirement is assigned to a single actor via an application or business service. The coordination of the use-cases that are included by the buy insurance use-case is assigned indirectly to the buy insurance business process.

Alignment with ArchiMate

The alignment of the proposed concepts and relations for requirements modeling involves, amongst other things, the following concerns:

- Re-use of existing concepts and notations if possible and appropriate, in order to keep ArchiMate lean and intuitive
- Definition of the relationship between the requirements concepts and the ArchiMate concepts

The first concern is addressed by re-using:

- The association relation to represent the relationships between stakeholders, concerns, assessments, and goals
- The aggregation relation to represent the decomposition relation between goals (including requirements)
- The realization relation to represent the means-end relation between goals (including requirements)

The second concern is addressed by defining the means-end relation between operational elements in the architecture (e.g., business or application services, business actors or processes, application components or functions) and requirements (including use-cases). This usage of the means-end relation is also represented using the realization relation from ArchiMate.

Example: PRO-FIT

This section presents an example to illustrate how the requirements engineering cycle and the language introduced in the previous sections can be applied in combination with TOGAF and ArchiMate. For brevity, we only elaborate the architecture vision and business architecture phases of the TOGAF ADM. The following case is used. As mentioned in Architecting Requirements – Way of Working, the architectural vision, business, information systems, and technology architecture are related to each other through problem frames. The business architecture refines the results from the architectural vision; the results from the business architecture phase are refined in the information systems architecture phase and the same holds true for the technology architecture.

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

Architecture Vision

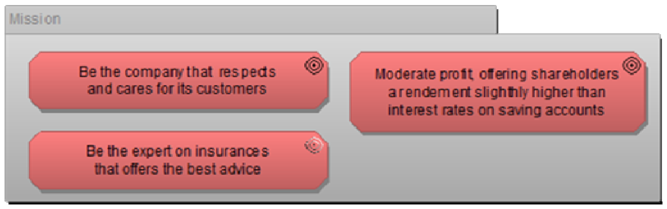
Mission
Be the company that respects and cares for its customers
Moderate profit, offering shareholders a rendement slightly higher than interest rates on saving accounts
Be the expert on insurances that offers the best advice

Figure 13: PRO-FIT's Mission

The **problem investigation** phase uses PRO-FIT's business plan as a starting-point for the elicitation of stakeholders and high-level business goals. Figure 13 depicts three business goals that are derived from PRO-FIT's mission.

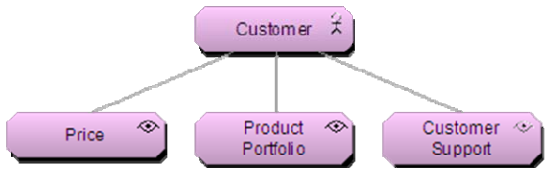

Figure 14: Customer Concerns

In addition, two main stakeholders are identified: the board of PRO-FIT and its customers. The board is concerned about profit, customer satisfaction, and innovation. PRO-FIT's customers are concerned about price, their product portfolio, and proper customer support. Figure 14 models the customer stakeholder and its concerns.

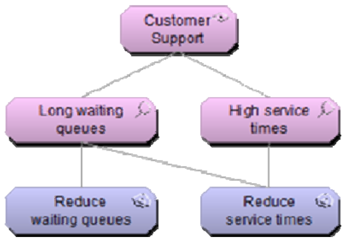

Figure 15: Assessment of Customer Support

The assessment of the “customer support” concern reveals that many customers experience long waiting queues and high service times when they call customer support to change their insurance portfolio. The goals “reduce waiting queues” and “reduce service times” are identified to address these assessments; see Figure 15.

Customers also experience a lack of insight in their insurance portfolio. This will be addressed by the goals “improve portfolio management”, “easy access to portfolio”, and “increase insight in portfolio”; see Figure 16.

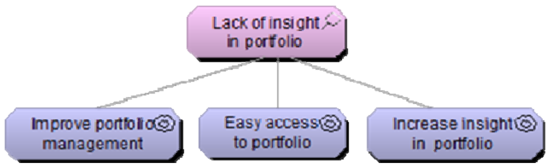
Lack of insight in portfolio
Improve portfolio management
Easy access to portfolio
Increase insight in portfolio

Figure 16: Addressing Assessments

Analysis of the preceding models shows that the goal “reduce service times” can be considered as a means to achieve the goal “reduce waiting queues”. Similarly, the goal “improve portfolio management” should realize the goals “easy access to portfolio” and “increase insight in portfolio”. These ends are again defined as sub-goals of “increase customer satisfaction”. The latter goal, including its sub-goals “improve claim handling” and “provide online services”, address concerns from the board (not modeled here). The aforementioned leads to the goal model as depicted in Figure 17.

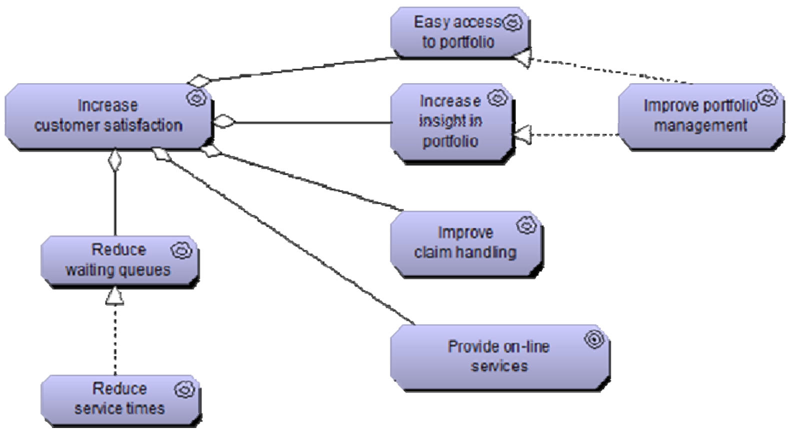

Figure 17: Analyzed Goal Model

The “investigate alternative solutions” phase uses goal refinement to decompose goals into more detailed goals and/or to find “means” to realize goals. Here, we focus on the refinement of the “improve portfolio management” goal. Figure 18 depicts that PRO-FIT should provide an online portfolio management service or personal management assistant for each customer to improve its portfolio a management.

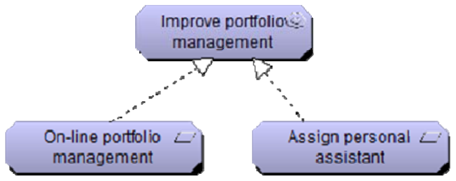
Improve portfolio management
On-line portfolio management
Assign personal assistant

Figure 18: Alternative Means to Some End

Figure 19 depicts the first architectural design effort. The requirement "online portfolio management" is realized through a "management service". The service is performed by the "customer" via a website, which forms the interface to the service. The realization of the alternative requirement, "assign personal assistant", also uses a management service. The major difference is that the customer and the management assistant perform the service together via phone.

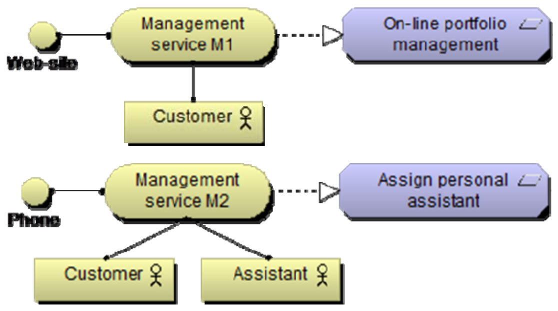

Figure 19: Architectural Designs that Realize Requirements

The above examples demonstrate how solution structures (the architectural elements) realize the early requirements. In this way, architecture elements can be traced back via the requirements they realize to the business goals and stakeholder concerns that motivated the introduction of these elements.

The "solution validation" phase should assess the quality of the alternative solutions in Figure 18. TOGAF recommends using a trade-off analysis to decide which direction should be favored in case alternatives are possible. A trade-off analysis can be implemented using the contribution relation from ARMOR. Figure 11 illustrated such analysis. From this analysis, PRO-FIT may decide to pursue online portfolio management as the favorite architectural vision.

Business Architecture

After agreeing upon an architectural vision, another cycle of the requirements architectural vision is now the problem under investigation.

The "problem investigation" phase is responsible for identifying new stakeholders and concerns to further refine the architecture vision. Figure 20 depicts a new stakeholder, the head of "operations", who is concerned about portfolio management, claim handling, and customer support. A closer look at claim handing reveals a number of problems. Customer information is often incomplete, claims get lost in internal mail, and claims archiving is sloppy. This leads to the goal to revise the claim handling process.

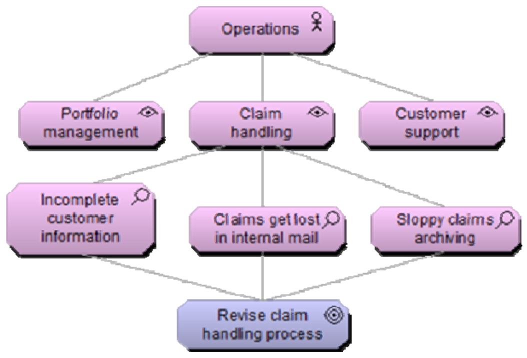
Operations
Portfolio management
Claim handling
Customer support
Incomplete customer information
Claims get lost in internal mail
Sloppy claims archiving
Revise claim handling process

Figure 20: A New Stakeholder, Concerns, Assessments, and High-Level Goal

The "investigate solution alternatives" phase is responsible for the refinement of the business goals. Figure 21 illustrates the decomposition of the goal "revise claim handling process" into the requirements to explicitly check customer information, to backup claims after claim submission, and to only evaluate claims larger than €150. Two alternatives solutions are identified for the requirement to check customer information; i.e., manually by an employee or automated using a CRM application. In this case, both alternatives are realized by the "claim accept service", which is implemented by a separate sub-process in the entire claim handling process. The requirement to back up claims is realized by means of a "backup service", which is invoked during the "claim submission" sub-process. The third requirement is realized by using a claim expert to perform the evaluation and by making claim evaluation an optional sub-process during claim handling.

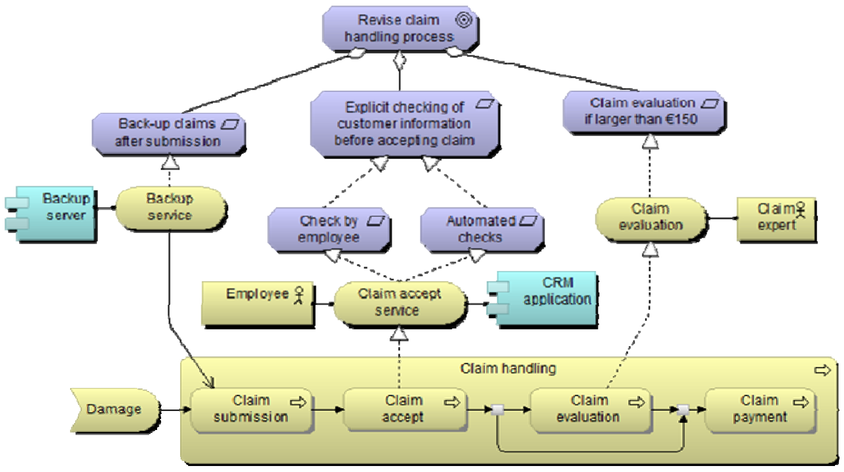

Figure 21: Business Architecture Design

In the claim handling example, no choice needs to be made among alternatives. However, the "solution validation" phase may apply other types of analyses to validate the solution. Examples of such analyses are: checking the completeness of the solution; i.e., whether all goals have been addressed, checking for conflicting or equivalent goals, and checking for circular realization relations between goals. Viewpoint techniques may be used to support these analyses. Techniques and Tools – Way of Supporting discusses several analysis techniques.

Techniques and Tools – Way of Supporting

A requirements model may grow quickly in subsequent RE cycles, describing many goals and requirements and many direct and indirect relations between them. Furthermore, the relations involving some goal or requirement may be distributed over different parts of the model. Therefore, tool support is needed to create, manipulate, and analyze requirements models.

Model creation and manipulation requires some editing tool. This tool should support both ArchiMate and the requirements extension proposed in this document.

Model analysis requires automated techniques to check, to assess, and to provide insight into a requirements model. In order to deal with complexity, the analysis of a requirements model may be separated into the analysis of distinct concerns or *aspects*. Each aspect stands for a particular *viewpoint* on requirements modeling, such as the goals and concerns per stakeholder, or the conflict and contribution relations per goal or set of goals. The representation of a requirements model from a particular viewpoint is called a *view*. Automated tool support is needed to generate and analyze views. Currently, the following categories of analyses are distinguished: completeness, consistency, trade-off between alternative solutions, and impact of change. These analyses build on the *traceability* between model elements, which is obtained by using the method and language for requirements management as described in the previous chapters.

In this White Paper, we present some extended modeling techniques to create, manipulate, and analyze requirements models.

Views

A view is defined by a selection of model elements and (indirect or derived) relationships between these elements. For example, views may be generated to show the goals and requirements that can be traced back to some stakeholder, all the stakeholders and concerns that are addressed (indirectly) by a particular goal, the highest-level goals that are supported by some business service or software application, etc. These views help the requirements analyst to provide insight in the requirements model. Furthermore, when combined with searching techniques, view generation may facilitate the re-use of parts of a requirements model. For example, the refinement of some goal into requirements and architectural elements may be re-used in the refinement of a related goal.

Given the possibility to generate views, the comparison of different views is a powerful technique to further analyze and increase insight into a requirements model. For example, Figure 22 depicts the comparison of two stakeholder views, where each view depicts the goals that address (indirectly) the concerns of a particular stakeholder. The blue elements are goals found in both stakeholder views. The presence of a goal in multiple views could be used as an indication for its importance.

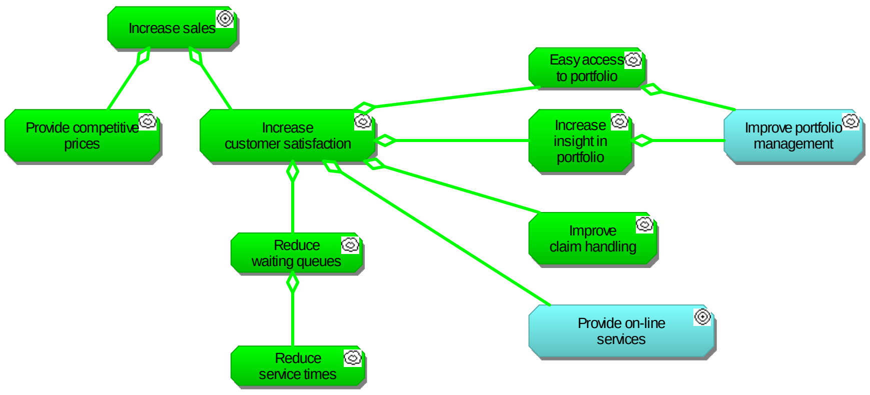

Figure 22: Comparison of Two Views

Completeness

Completeness of a requirements model means that each goal or requirement is traceable from its origin to its (alternative) realization(s). A goal should be traceable *backwards* to one or more stakeholders via the assessment(s) and concern(s) it addresses. A goal should be traceable *forwards* to sub-goals or requirements via a decomposition or means-end relation. For example, the goal "revise claim handling process" in Figure 21 is decomposed into three sub-goals.

As a specialization of the notion of goal, a requirement should be traceable forwards to sub-goals or requirements via a decomposition or means-end relation (inherited from goal), or to one or more architectural elements that realize the requirement via a means-end relation. A model is considered complete with respect to a requirement, if it can be traced forwards (directly or indirectly) to a behavior element that realizes the required functionality and/or a structural element that performs the behavior element. For example, the requirement "backup claims after submission' in Figure 21 is realized by the "backup service" and the application component "backup server" (via the assignment relation).

Consistency

Various techniques can be distinguished and used to analyze the consistency or correctness of a requirements model. Some examples are presented below.

Goal cycle analysis detects goals that mutually depend on each other, directly or indirectly, via a realization or decomposition relation. Goal cycles may be introduced during goal elicitation when the problem structure is not entirely clear yet.

Figure 23 (top part) depicts an example of a goal cycle. In the "customer support" stakeholder view "increase customer satisfaction" is a means towards the end "reduce service times". But in the "board" stakeholder view "reduce service times" is a means towards the end "increase customer satisfaction". This goal cycle may require a new assessment of some stakeholder concerns, because the initial problem is not clear. For example, a new assessment may learn that "customers call customer support because they cannot modify their portfolio by themselves". This allows one to resolve the goal cycle by introducing the goal "improve portfolio management" to realize both other goals, and possibly using a contribution relationship to model how these goals influence each other (see bottom part of Figure 23).

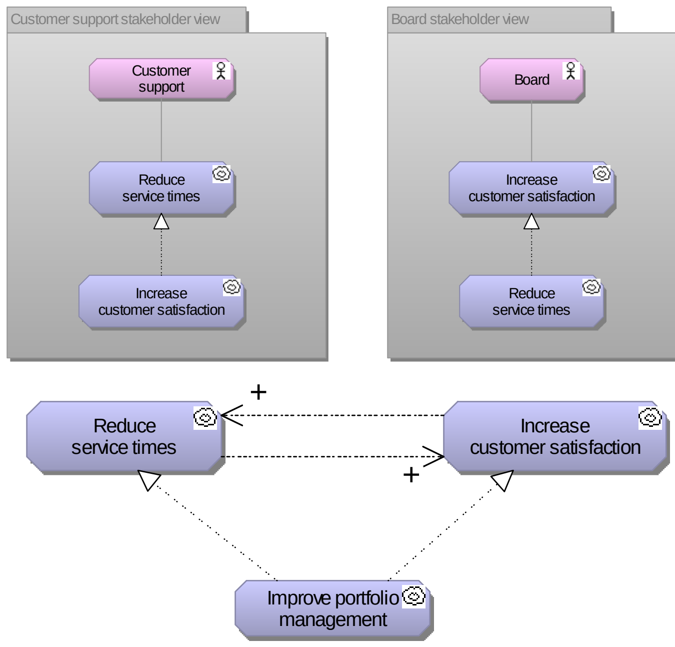
Customer support stakeholder view
Customer support
Reduce service times
Increase customer satisfaction
Board stakeholder view
Board
Increase customer satisfaction
Reduce service times
+
Reduce service times
Increase customer satisfaction
+
Improve portfolio management

Figure 23: Goal Cycle

Goal equivalence analysis detects goals that correspond syntactically and/or semantically. For example, during elicitation equivalent goals may be identified using (slightly) different names and specifications. The detection of semantic equivalences requires the use of a vocabulary in combination with a technique to describe the semantics of the words and sentences that are used to describe goals; e.g., by using some ontology.

Goal conflict analysis detects indirect conflict relations between goals. A conflict occurs when the realization of one goal is sufficient to obstruct the realization of another goal, and *vice versa*. Conflicting goals are often found during goal refinement and the representation of these conflicts is supported by the requirements modeling language as proposed in this document. For example, Figure 11 depicts a conflict between “reduce CS staff” and “assign personal assistant”. From this conflict one can derive that a goal G1 that depends on the former goal is in conflict with any goal G2 that depends on the latter goal.

Trade-Off Between Alternative Solutions

During goal refinement alternative solutions may be identified. These alternatives are represented using the realization relation. Figure 24 revisits the example of Figure 11. Based on the contribution relation, quantitative evaluation techniques can be used to decide among alternative solutions. For example, one may calculate the weighted average of the contribution of each alternative solution to multiple (typically soft) goals. In this example, the solution “online portfolio management” contributes on average higher to the soft goals “increase customer satisfaction” and “improve CS efficiency” than the alternative solution “assign personal assistant”.

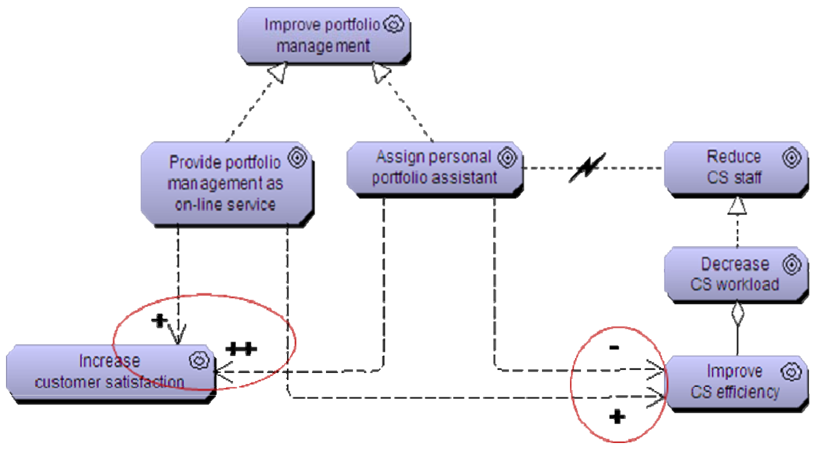
Improve portfolio management
Provide portfolio management as on-line service
Assign personal portfolio assistant
Reduce CS staff
Decrease CS workload
Increase customer satisfaction
Improve CS efficiency
+
++
-
+

Figure 24: Trade-Off Analysis

Impact of Change

Traceability enables impact of change analysis. For example, in case of changing legislation, certain high-level goals that address the compliance to this legislation may change. By tracing the relationships from these goals to the requirements and the architectural elements that realize these requirements, someone can determine which business and IT elements are affected by this change. Furthermore, in addition to forwards tracing, impact of change can be traced in the "backwards direction". For example, when some IT requirement cannot be met, the business process that imposes this requirement cannot be realized. As a consequence, the organization may not be able to offer some of its business services. Following these dependencies further, one may assess the impact of the IT requirement on the business goals and strategy of the organization.

Conclusions

In this White Paper we have presented an extension to TOGAF and ArchiMate for requirements management. This extension constitutes a practical way of *thinking*, *working*, *modeling*, and *supporting*.

We have demonstrated how enterprise architecture design and requirements management can be related through the RE cycle. This contributes to a better grip on the architecture development process, consisting of:

- *Better motivation and foundation* of architecture models and design choices
- *Better insight* in architecture models due to enhanced traceability between requirements elements and architecture elements; this also enables *clearer communication* of architecture models
- Additional and *better analysis techniques* for consistency, completeness, trade-off between alternative solutions, and impact of change
- A *flexible method* for requirements management that can be configured to fit the organization at hand using existing RE techniques
- A *generic method and modeling language* that can be applied to enterprise architecture design, but might also be applicable to business process design and functional design

The integration of enterprise architecture design and requirements management also contributes to the "closing" of the business-IT gap. Architecture models are aligned to the vision, mission, and strategy of the organization by eliciting goals from stakeholder concerns and the organization's business plan and by refining these goals into products, business services, business processes, and software applications that realize these goals.

References

[1] Goal Decomposition and Scenario Analysis in Business Process Re-engineering, A.I. Antón, W.M. McCracken, & C. Potts, Proceedings of the 6th International Conference on Advanced Information Systems Engineering, pp94–104, 1994.

[2] No Silver Bullet: Essence and Accidents of Software Engineering, F.P. Brooks, *IEEE Computer*, 20(4):10–19, 1987.

[3] Goal-directed Requirements Acquisition, A. Dardenne, A. van Lamsweerde, & S. Fickas, Science of Computer Programming, 20(1-2):3–50, 1993.

[4] Prioritizing Requirements, D. Firesmith, Journal of Object Technology, 3(8):35–47, 2004.

[5] Techniques for Requirements Elicitation, J.A. Goguen & C. Linde, Requirements Engineering, 93:152–164, 1993.

[6] ArchiMate 1.0 Specification, The Open Group, published by Van Haren Publishing, 2009.

[7] The Chaos Report, The Standish Group, 1994.

[8] TOGAF and ArchiMate: A Future Together, H. Jonkers, E. Proper, M. Turner, White Paper (W192), submitted to The Open Group.

[9] A Goal-Oriented Requirements Modeling Language for Enterprise Architecture, D.A.C. Quartel, W. Engelsman, H. Jonkers, & M.J. van Sinderen, Proceedings of the 13th IEEE International EDOC Enterprise Computing Conference, Auckland, New Zealand, pp3-13, September 2009.

[10] TOGAF Version 9, The Open Group, published by Van Haren Publishing, 2009.

[11] Goal-Oriented Requirements Engineering: A Guided Tour, A. van Lamsweerde et al, Proceedings of the 5th IEEE International Symposium on Requirements Engineering, p249, 2001.

[12] Enterprise Architecture: The Issue of the Century, J.A. Zachman, Database Programming and Design, 10:44–53, 1997.

[13] A Framework for Information Systems Architecture, J.A. Zachman, IBM Systems Journal, 38(2/3):454–470, 1999.

About the Authors

Wilco Engelsman is a Research Consultant at BiZZdesign. In this capacity, he is involved in the company's new developments in the areas of requirements engineering and enterprise architecture; he participates in multi-party research projects, as well as in consultancy for customers.

Henk Jonkers is a Senior Research Consultant at BiZZdesign. In this capacity, he is involved in the company's new developments in the areas of business process engineering and enterprise architecture. He participates in multi-party research projects, as well as in consultancy for customers. Henk was one of the main developers of ArchiMate and an author of the ArchiMate 1.0 Specification. He is TOGAF 8 and 9 certified.

Dick Quartel is a Senior Researcher at Novay. In this capacity, he is involved in research, consultancy, project management, and project development. The mission of Novay is to facilitate innovation enabled by ICT in companies and public organizations. Before joining Novay, Dick worked as an assistant professor at the University of Twente. His expertise and interests include enterprise and service-oriented architecture, business process (re)design, model-driven development of services and software, service composition and interoperability, and requirements modeling. Dick has worked in several national and European projects.

About The Open Group

The Open Group is a vendor-neutral and technology-neutral consortium, whose vision of Boundaryless Information Flow™ will enable access to integrated information within and between enterprises based on open standards and global interoperability. The Open Group works with customers, suppliers, consortia, and other standards bodies. Its role is to capture, understand, and address current and emerging requirements, establish policies, and share best practices; to facilitate interoperability, develop consensus, and evolve and integrate specifications and Open Source technologies; to offer a comprehensive set of services to enhance the operational efficiency of consortia; and to operate the industry's premier certification service, including UNIX® system certification. Further information on The Open Group can be found at www.opengroup.org.