



# TUM

TECHNISCHE UNIVERSITÄT MÜNCHEN  
INSTITUT FÜR INFORMATIK

## Mini-Guideline to Requirements Engineering

Birgit Penzenstadler

TUM-I126

Technische Universität München  
Institut für Informatik

# Technischer Bericht

# Mini-Guideline to Requirements Engineering

Birgit Penzenstadler, Software & Systems Engineering, penzenst@in.tum.de

**Abstract**—Requirements engineering (RE) is an essential part of any (software) development project. It captures and models the domain as well as the needs of the stakeholders and systematically derives the requirements and constraints until the information is concrete enough to design a system that suits the needs. Thus, RE provides a thorough understanding of the future system and its environment. RE helps to find out what your customer wants and actually needs, and provides the foundation for designing the system that suits these requirements.

This guideline lends a hand in getting started with performing requirements engineering in a software development project. It explains how to proceed with RE, what information to capture, and how to document the results. The approach is based on a domain-independent content model for artefact-oriented requirements engineering.

## I. MOTIVATION AND BACKGROUND

When customers issue a contract, they want to see fast progress and results. At the same time, most developers are eager to start designing straight away having a lot of ideas in mind that might help their customer.

The problem is: the later requirements are added (because they were missing in the first iteration) or changed (because there were conflicts or inconsistencies), the more expensive your system's development — likely to blast your project's budget calculation and imposing delay. A significant number of software development projects fails due to insufficient requirements engineering. So before you put effort in designing a system that will probably not suit your customer's needs, better make sure that you have elicited the right requirements and that they are consistent. To be prepared for this, you learn how to perform requirements engineering.

Our course on requirements engineering for Master students covers a wide range of topics, perspectives, and methods. To integrate these topics and to provide a common backbone as basis, we provide an artefact model that consists of a content model and the respective structures for the items in the content model. In Fig. 1, you see how the artefact model is composed by content and structure and how they relate to each other.

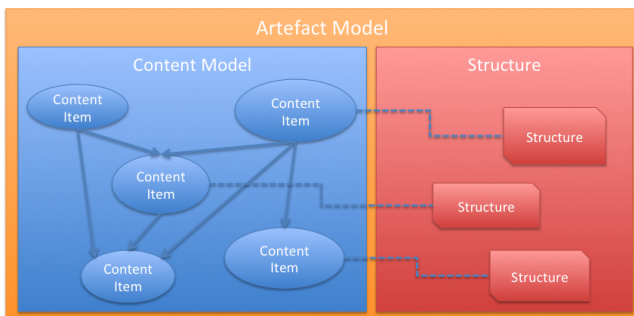


Fig. 1. Artefact Model with Content and Structure.

We consider a content model a description of the general content, to be created in requirements specifications. The model is independent of a document structure or data sets and especially independent of chosen description techniques for representing the content. We provide suggestions for the description techniques as a starting point.

## II. OVERVIEW OF RE-PHASES

In general, we distinguish the following phases in the requirements engineering process:

- Elicitation: find out about the information
- Analysis: think about it and structure it
- Specification & Documentation: write it down
- Validation & Verification: check back with stakeholder

Every requirement runs through these phases. Each phase is realised in tasks that can also span across phases. To carry out the tasks, there is a number of methods that can be used. Figure 2 shows the relation between the terms illustrated by an example.

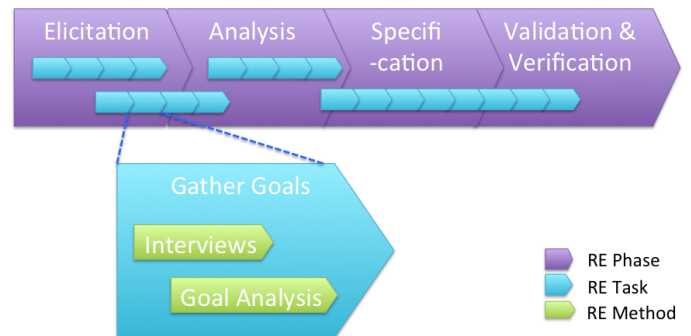


Fig. 2. Overview of RE Phases, Tasks, and Methods.

In our approach, the information to be captured in RE is defined in a Content Model. It serves as a checklist for the different aspects to be considered. Furthermore, the Content Model is a backbone to structure the specific tasks to be carried out during the requirements engineering phases. These tasks, for example, “gather goals”, are supported by methods and techniques. The results from these activities are captured in Content Items. The content items can be grouped according to the information needs of a specific stakeholder.

Artefacts are content items with a predefined structure. A complete artefact model is therefore too complex to explain within this mini guideline, therefore we only ask you to remember the definition. We choose such an artefact-oriented approach because it allows for more flexibility in the process that is carried out with respect to distributed work and scheduling that defines strictly sequential, fine-grained process steps.

### III. CONTENT MODEL

The following description of the domain-independent RE content model in Fig. 3 is a first sketch of contents to be covered by an artefact model (see Fig. 1). Thereby, an artefact is a deliverable that is produced, modified, or used by a sequence of tasks that have value to a role (taken by a stakeholder). Artefacts are subject to quality assurance and version control and have a specific type. The content model is hierarchically structured into content items that define single areas of responsibility and that are the output of a single task. At its lowest level of decomposition, each content item encompasses [MPKB10]:

- 1) Concepts: a concept defines the elements and their dependencies on domain-specific description techniques used to represent the concern of a content item. Concepts have a specific type and can be decomposed to concept items. The latter differentiation is made if different items of a concept can be described with different techniques.
- 2) Syntax: the syntax defines a concrete language or representation that can be chosen for a specific concept.
- 3) Method: The method (or task) describes the sequence of steps that is performed in order to make use of a concept.

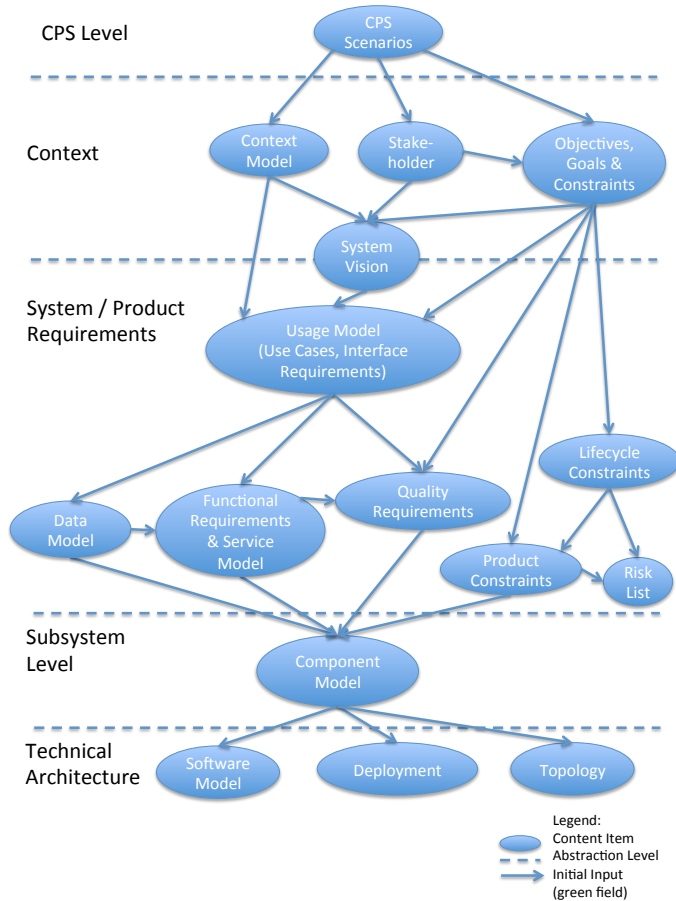


Fig. 3. Domain-independent RE Content Model.

The arrows of Fig. 3 indicate that the contents are initially (supposing green field development) derived from the preced-

ing contents. We are aware that green field development is rarely the case and the model does not require green field. However, it is easier to illustrate and explain in top-down order. There are more influences that are not depicted in order not to overload the graphic.

#### A. Abstraction Levels

The model is structured by means of four abstraction levels: Environment, System, Logical Architecture and Technical Architecture. Requirements Engineering focuses on the upper two levels. The two lower abstraction levels are included to facilitate integration with design and to enable bottom-up promotion of technical constraints.

- Environment / Context (Operational and Business): The Environment is described independently of a concrete system.
- System / Product Requirements, Functional Architecture: On this level, the interaction between users and system is described within the problem domain (black box).
- Subsystem Level, Logical Architecture: On this level, the system is described as white box in terms of structure and behaviour while still not distinguishing between hardware or software.
- Technical Architecture: This level describes the realisation of the system in hardware and software.

#### B. Content Items

The descriptions in Tables I, II, III, and IV are short definitions of the content items depicted in Fig. 3.

TABLE I  
CONTENT ITEMS OF THE OPERATIONAL ENVIRONMENT

| Content Item                      | Description                                                                                                                                                                                                                                                                                               |
|-----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Domain Model                      | A system-independent description of the domain and the operational and business environment.                                                                                                                                                                                                              |
| Stakeholder                       | A description of individuals or organisations that are related to the project.                                                                                                                                                                                                                            |
| Objectives, Goals and Constraints | An objective is a major statement of intent to be achieved by a project, while a goal is a more concrete, prescriptive statement of intent of one or more stakeholders. Constraints are restrictions of any type. Objectives and Goals are represented by same Concepts; we refine Objectives into Goals. |

The content items in the lower two abstraction levels are usually not elaborated during requirements engineering. They are defined here to facilitate relating to the design phase of the development.

TABLE II  
CONTENT ITEMS OF THE SYSTEM LEVEL

| Content Item          | Description                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| System Vision         | An outline of the system's capabilities and characteristics.                                                                                                                                                                                                                                                                                                                                                                                  |
| Usage Model           | A description of observable system behaviour and interaction from the viewpoint of the user (use cases, scenarios).                                                                                                                                                                                                                                                                                                                           |
| Functions / Services  | The single units of functionality that realise the behaviour specified in the usage model. Usage Model and Functions (hierarchy) are both views on black box behaviour from different perspectives (domain vs. technical).                                                                                                                                                                                                                    |
| Quality Requirements  | A description of properties and conditions for structure or behaviour of the system by use of measures. Quality Requirements and Constraints refer to the same quality model. There are several views for quality that are determined by the stakeholder who issues a quality requirement, for example, user, maintainer, legislation, etc. One quality attribute or also one quality requirement can thereby also have several stakeholders. |
| Data Model            | The information processed during the execution of functions.                                                                                                                                                                                                                                                                                                                                                                                  |
| Product Constraints   | The quality-independent restrictions on functional, logical, and technical architecture.                                                                                                                                                                                                                                                                                                                                                      |
| Lifecycle Constraints | The restrictions and agreements concerning development process, release, integration, and maintenance. Product Constraints and Lifecycle Constraints are based on the same concepts but have different stakeholders.                                                                                                                                                                                                                          |
| Risk Report / List    | The description of potential problems related to requirements or project.                                                                                                                                                                                                                                                                                                                                                                     |

TABLE III  
CONTENT ITEMS OF THE COMPONENT LEVEL

| Content Item    | Description                                                                                                                                                 |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Component Model | The structure and behaviour of the logical components that realise the functions. Thereby, a component can again be perceived as and treated like a system. |

#### IV. TASKS AND METHODS

The items of the content model are developed iteratively across the phases mentioned in Sec. II and influence each other. Nevertheless, to provide an order of describing a possible initial development of requirements, we give an example of how the first steps can be conducted. We illustrate the steps using the example of a business information system for the fictitious company “Alpine Adventure Tours” (AAT) that supports their course bookings, customer management, procurement, and reporting.

##### A. Starting point

The most common starting point for requirements engineering is a customer who issues the desire for a specific software solution. In the content model, this is one of the stakeholders. They have a system vision. From your customer you get a first version of the system vision and you get more stakeholders to talk to.

TABLE IV  
CONTENT ITEMS OF THE TECHNICAL LEVEL

| Content Item   | Description                                             |
|----------------|---------------------------------------------------------|
| Topology       | The hardware and communication outline.                 |
| Software Model | The packaging and communication of software components. |
| Deployment     | The mapping of software to hardware and integration.    |

##### B. Finding the Stakeholders

One major pitfall for requirements engineering is to have an incomplete list of stakeholders (content item on the Operational Environment level in Fig. 3). This pitfall can easily be avoided by systematically gathering the people who have a potential interest in the system.

For every stakeholder, choosing the right vocabulary to reach out to them is important. Whether or not a stakeholder has technical knowledge, is involved with marketing, or represents legal concerns — make clear that their points of view are relevant and their requirements are taken into account. Important common types of stakeholders that should be checked for every system are:

- Customer
- User Groups
- Marketing & Sales
- Legislation
- Developers (Hardware and Software)
- Help Desk, Technical Support & Maintenance

Figure 4 shows the most important stakeholders of the Alpine Adventure Tours example.

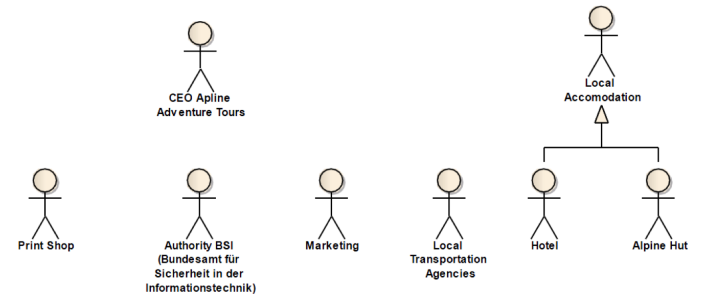


Fig. 4. Stakeholders of Alpine Adventure Tours.

A helpful reference for further reading is the taxonomy of stakeholders by Ian Alexander, called onion model [Ale05].

### C. Agreeing on System Vision & Scope

All stakeholders should agree on a common system vision and scope. This implies that the system vision (content item on the System level in Fig. 3) has to be understood by every stakeholder involved — including non-technicians. We advise to use a so-called rich picture [MH98] as illustration. A rich picture captures the key elements of the system vision in (self-explanatory or labelled) icons and depicts their interrelation. Figure 5 shows a rich picture of the system vision of AAT. Rich pictures have proven especially useful as basis for

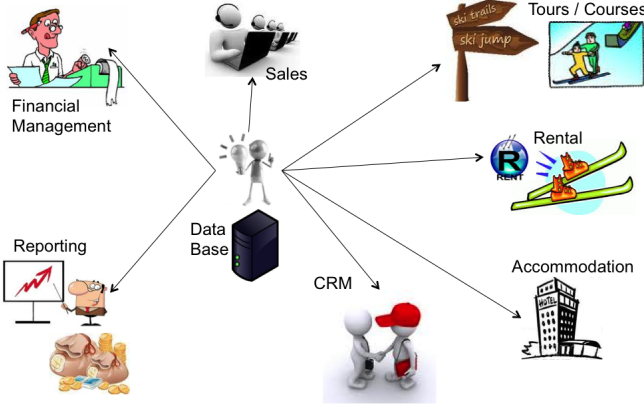


Fig. 5. System Vision of Alpine Adventure Tours.

discussion in workshops and meetings. In documentation, i.e. after the stakeholders have agreed on one version of the rich picture, it should be accompanied by an explaining paragraph in natural language.

### D. Gathering Goals, Usage Model, and Constraints

From the stakeholders, we gather the goals (content item on the Operational Environment level in Fig. 3). Goals can be business goals, market goals, functional, quality, or technical goals. The two most common forms of documenting goals are natural language text and goal trees [vL01]. The latter have the advantage of showing the relations between the goals.

Example goals from AAT are:

- Competition with other skiing regions demands high customer satisfaction at low prices.
- Customers must get the business services from one hand.
- Increase of customer satisfaction by reduction of customer complaints.
- Fast market expansion and branding the business image in new markets with collaboration of existing local companies.
- Providing the best service to customers.
- Achieving market lead in skiing courses at region “Zugspitze”.

Apart from the stakeholders, there are other sources of information, usually documents. Some important information sources are:

- Legacy systems and user documentation
- Laws, standards, and regulations
- Customer complaints, unintended uses

To turn the goals into concrete requirements, they have to be refined. The goals that refer to the services to be provided by the system are refined into descriptions of usage behaviour. This often requires additional input from both stakeholders and other information sources. The results of this analysis are captured in the content item Usage Model (on the System level in Fig. 3). Usage can be captured either in service descriptions or in use cases and scenarios. The usage model is a black box specification of the behaviour that hides any realization detail. Figure 6 shows the use case overview of AAT and Figure 7 depicts a detail of the scenario derivation.

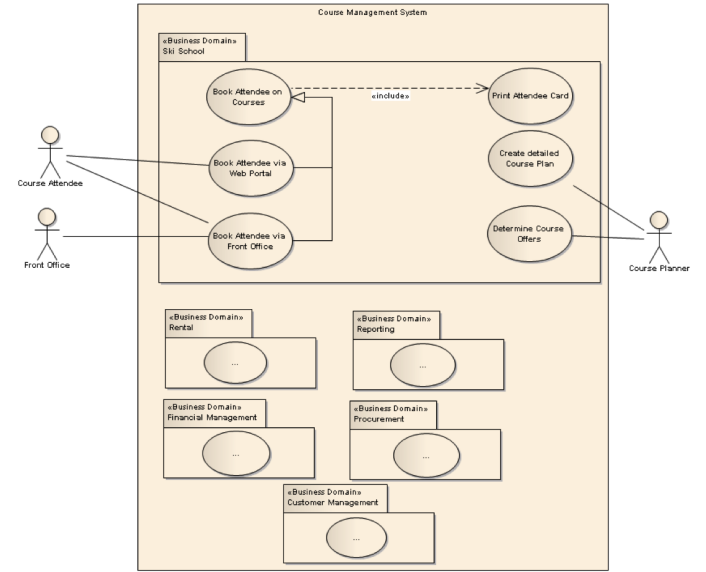


Fig. 6. Use Case Overview of Alpine Adventure Tours.

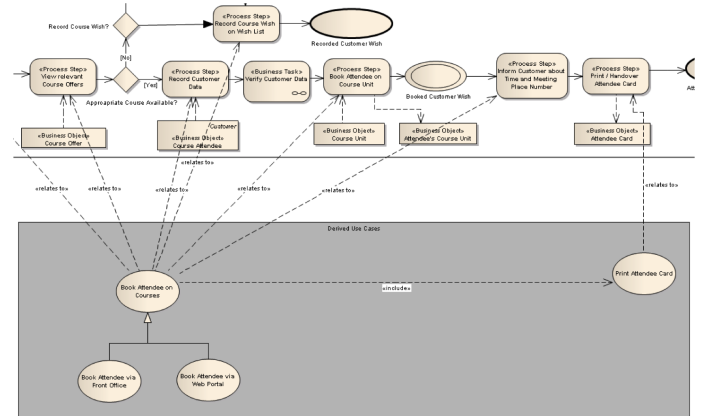


Fig. 7. Scenario Detail of Alpine Adventure Tours.

Constraints describe restrictions that arise from the business context (like management or laws) or from the system’s operational environment (like hardware constraints). Both types have to be listed with references to their original source.

### E. Documenting Requirements

On the basis of the goals, the usage model, and the constraints, the requirements are derived and documented. Write your requirements SMART — specific, measurable, attainable, realisable, and time bounded (objective must be achieved by a specific date in the project plan).

Your requirements need a number of attributes:

- ID, version, and configuration (if applicable)
- Origin, author, and responsible
- Rationale and tracing to related requirements or artefacts
- Priority and status

A popular template for requirements documentation is the Volere Template [RR06], see Fig. 8.

|                        |                           |                   |
|------------------------|---------------------------|-------------------|
| Requirement #:         | Requirement Type:         | Event/use case #: |
| Description:           |                           |                   |
| Rationale:             |                           |                   |
| Source:                |                           |                   |
| Fit Criterion:         |                           |                   |
| Customer Satisfaction: | Customer Dissatisfaction: |                   |
| Dependencies:          | Conflicts:                |                   |
| Supporting Materials:  |                           |                   |
| History:               |                           |                   |

  
Copyright © Atlantic Systems Guild

Fig. 8. Volere Template for Requirements.

### F. Quality Assurance and Acceptance

There are two major stages of review to perform — one by your colleagues (internal) and one by your customer (external). The internal review checks understandability, completeness, consistency, precision, correctness, traceability, and changeability.

The external review by the customer performs the same checks plus the decision whether the requirements specification actually describes what they want. Your goal is agreement with the customer and acceptance of your requirements specification — and the preceding internal review is the basis.

### G. Managing Requirements

Your requirements are likely to be changed by the customer and other stakeholders during development. Therefore, it is important to establish a proper change management process that keeps your requirements specification consistent and their change history traceable. Best practice is to issue change requests that are decided on. These decisions, as well as other decisions taken during development, should be documented for future reference. A helpful template for decision documentation is provided by Tyree [TA05]. It captures issue, decision, status, assumptions, constraints, implications, related decisions and requirements. The extent of a template should be adapted to the project settings — but: document the decisions.

## V. CONCLUSION

This guideline gives concrete steps on the way to accomplish your first requirements engineering project. We welcome feedback to further improve it for the future.

## ACKNOWLEDGEMENTS

Thanks a lot to Veronika Bauer, Maximilian Junker, and Mario Gleirscher for input and helpful feedback on earlier versions of this guideline.

## REFERENCES

- [Ale05] Ian F. Alexander. A taxonomy of stakeholders: Human roles in system development. *International Journal of Technology and Human Interaction*, 1(1):23–59, 2005.
- [MH98] Andrew Monk and Steve Howard. The rich picture: a tool for reasoning about work context. *Interactions*, 5(2):21–30, 1998.
- [MPKB10] Daniel Mendez, Birgit Penzenstadler, Marco Kuhrmann, and Manfred Broy. A meta model for artefact-orientation: Fundamentals and lessons learned in requirements engineering. In *Proc. of the 13th International Conference on Model Driven Engineering Languages and Systems*, 2010.
- [RR06] James Robertson and Suzanne Robertson. Volere: Requirements specification template, 2006. <http://www.volere.co.uk/>.
- [TA05] Jeff Tyree and Art Akerman. Architecture decisions: Demystifying architecture. *IEEE Softw.*, 22:19–27, March 2005.
- [vL01] Axel van Lamsweerde. Goal-oriented requirements engineering: A guided tour. In *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering*, page 249, 2001.