Software Engineering for Software-Intensive Systems: IV Requirements

Assistant Professor Dr. Holger Giese
Software Engineering Group
Room E 3.165
Tel. 60-3321
Email: hg@upb.de
Outline

I  Introduction
II  Foundations
III  The Development Life Cycle
IV  Requirements
V  Analysis & Design
VI  Implementation
VII  Verification & Validation
VIII  Summary and Outlook
IV Requirements

IV.1 Requirements Engineering

IV.2 Requirement Specification
IV.3 Approach: SysML
IV.4 Approach: Goal-oriented
IV.5 Discussion & Summary
IV.6 Bibliography
Why are Requirements so important?

**Distribution of software errors**
- Requirements: 82%
- Design: 13%
- Code: 1%
- Other: 4%

**Cost of rectifying errors**
- Requirements: 56%
- Design: 27%
- Code: 7%
- Other: 10%

[Cooling2002]
What is Requirement Engineering?

- **Requirement Engineering (RE)** is the science and discipline concerned with analyzing and documenting requirements.

- **requirement.**
  1. A condition or capability needed by a user to solve a problem or achieve an objective.
  2. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.
  3. A documented representation of a condition or capability as in (1) or (2).

[IEEE-Std-610.12-1990]
Requirement Characteristics

i. **Type**: the source and contractual applicability

ii. **Application**: the object of a requirement

iii. **Compliance Level**: the depth of compliance mandated for a requirement

iv. **Priority**: relative importance of a requirement
i. Requirement Type

- **Primary requirements**: Sources:
  - Contract or pre-contract document
  - established by management or marketing

- **Derived requirements**:
  - Derived from a primary requirement
  - Derived from a higher level derived requirement

[Thayer&Dorfman1997, p. 29]
ii. Requirement Application

- **Product parameter**: applies to a product or service to be developed
  - **Qualitative** – not directly measurable
    - children (derived requirements) refine which provide quantifiable criteria should be met
  - **Quantitative** – measurable
    - children (derived requirements) can be generated for the purpose of specifying particular approaches to meet this measurable requirement

- **Program Parameter**: activities associated with enabling the creation of the product/service
  - **Task**: effort to be performed
  - **Compliance Evaluation**: methodology for measuring compliance
  - **Regulatory**: administrative elements

[Thayer&Dorfman1997, p. 29]
iii. Requirement Compliance Level

- **Mandatory**: must be implemented
- **Guidance**: desirable that it be implemented
- **Information**: non-binding statements which significantly influence the context, meaning, and understanding of other requirements

[Thayer&Dorfman1997, p. 29]
iv. Requirement Priority

- Characterize the *relative* importance of a requirement
  - Basis for trade studies

- Unlike the other characteristics the priority depends on company needs

[Thayer&Dorfman1997, p. 29]
A Good Set of Requirements is …

- Correct
- Unambiguous
- Complete
- Consistent
- Ranked for importance and/or stability
- Verifiable

[Thayer&Dorfman1997]
[IEEE Std. 830-1993]
[IEEE Std. 830-1998]
System vs. Software

- **System Requirement Engineering** is the science and discipline concerned with analyzing and documenting system requirements.
  
  *origin*: user needs

- **Software Requirement Engineering** is the science and discipline concerned with analyzing and documenting software requirements.
  
  *origin*: system requirements and/or specification

**BUT**: For software-intensive systems software people should be involved in the elicitation of the system requirements!
The Context of Requirement Engineering

Stakeholder needs

Existing systems information

[Kotonya&Sommerville1998]
## The Inputs and Outputs

[Kotonya&Sommerville1998]

<table>
<thead>
<tr>
<th>Input or output</th>
<th>Type</th>
<th>Descriptionlinha</th>
</tr>
</thead>
<tbody>
<tr>
<td>Existing system information</td>
<td>Input</td>
<td>Information about the functionality of systems to be replaced or other systems which interact with the system being specified</td>
</tr>
<tr>
<td>Stakeholder needs</td>
<td>Input</td>
<td>Descriptions of what stakeholders need from the system to support their work</td>
</tr>
<tr>
<td>Organisational standards</td>
<td>Input</td>
<td>Standards used in an organisation regarding system development practice, quality management, etc</td>
</tr>
<tr>
<td>Regulations</td>
<td>Input</td>
<td>External regulations such as health and safety regulations with which the system must comply</td>
</tr>
<tr>
<td>Domain information</td>
<td>Input</td>
<td>General information about the application domain of the system</td>
</tr>
<tr>
<td>Agreed requirements</td>
<td>Output</td>
<td>A description of the system requirements which is understandable by stakeholders and which has been agreed by them</td>
</tr>
</tbody>
</table>
RE Elements

(1) Requirement Elicitation
(2) Requirement Analysis
(3) Requirement Specification
(4) Requirement Validation
(5) Requirement Management
(1) Requirement Elicitation

- **Requirement Elicitation**: the process through which the customer and developer discover, review, articulate, and understand the users’ needs and constraints on the software and development activities.

- Requirements elicitation is about discovering what requirements a system should be based upon.

- This doesn’t involve just asking stakeholders what they **Want**. It requires a careful analysis of:
  - The organisation
  - The application domain
  - Organisation processes where the system will be used

- To determine what the stakeholders **Need**.
Elicitation Process

- Establish objectives
  - Business goals
  - Problem to be solved
  - System constraints

- Understand background
  - Organisational structure
  - Application domain
  - Existing systems

- Organise knowledge
  - Stakeholder identification
  - Goal prioritisation
  - Domain knowledge filtering

- Collect requirements
  - Stakeholder requirements
  - Domain requirements
  - Organisational requirements
Stakeholders

Who are they?

- Anyone with a stake in creating or using a new system
  - Hands-on users
  - Their managers
  - The system administrator
  - The client (system owner)
  - The requirements engineer
  - Other Engineers
  - Software designers
  - Programmers

Stakeholders’ backgrounds vary: they may:

- come from different departments
- be trained in different disciplines
- have different (possibly conflicting) goals
- be unwilling to consider other stakeholders’ goals
- have more or less political influence over requirements decisions
Uncovered Knowledge

- Application domain knowledge
  - E.g. knowledge about airport systems
- Problem context knowledge
  - E.g. knowledge about Dallas Airport
- Problem knowledge
  - E.g. knowledge about Dallas’s baggage-handling system
- Stakeholders needs and work processes to be supported
Requirements Elicitation Techniques

- Interviews
- Questionnaires
- Examination of documentation
  - Standards
  - Systems manuals
  - Statement of requirements
- Prototyping
- Contextual Design
- Conversation and interaction analysis
(2) Requirement Analysis

- Requirement Analysis: the process of analyzing the customers’ and users’ needs to arrive at a definition of the requirements.

- requirements analysis.
  (1) The process of studying user needs to arrive at a definition of system, hardware, or software requirements.
  (2) The process of studying and refining system, hardware, or software requirements.

[IEEE-Std-610.12-1990]
Analysis & Negotiation

- The analysis has to establish an agreed set of requirements which are **complete**, **consistent**, and **unambiguous**. Such a set can be used as the basis for systems development.

- **Negotiation:** Stakeholders often disagree over requirements. Therefore they need to negotiate to try to reach agreement.
(3) Requirement Specification

- Requirement Specification: the development of a document that clearly and precisely records each of the requirements of the software.

see IV.2
(4) Requirement Validation

Requirement Validation: the process of ensuring that the requirement specification is in compliance with the user needs, system requirements, conforms to document standards, and is an adequate basis for the architectural design.

see Chapter VII
(5) Requirement Management

- Requirement Management: the planning and controlling of the requirements elicitation, specification, analysis, and verification activities.

[Thayer&Dorfman1997]
Requirements for Complex Systems

- A system of any but the smallest size will be decomposed into a hierarchy of elements (partitioning):

- This is reflected at the requirement level by:
  1. **Allocation**: assigning requirements to elements
  2. **Flowdown**: requirements which respond to the allocated highel level requirements
  3. **Traceability**: keep track of the dependencies

[Thayer&Dorfman1997]
Effort Distribution for Complex Systems

- For complex systems due to allocation and flowdown the requirement engineering continues during the design (and implementation) phase.

Naïve view

![Effort Distribution Diagram](coolman2002)
Requirements vs. Design

Distinction:
- Design solution: **HOW** to achieve something
- Requirements: **WHAT** to achieve

Two cases where a requirements and design solutions are mixed up:
- The customer mandates a design solution as a requirement
  - Design solution (HOW): “provide a database for X”
  - Requirements (WHAT): “capabilities for navigation and sort for X”
  - Otherwise:
    - Restrict design space
    - Risk to miss requirements: ask WHY!
- A derived requirement which is actually a design solution and no requirement
  - See allocation and flowdown
  - Often alternation of requirements analysis and design
  - One person’s design is the next person’s requirements

[Thayer&Dorfman1997]
IV.2 Requirement Specification

IV.1 Requirements Engineering

IV.2 Requirement Specification

IV.3 Approach: SysML

IV.4 Approach: Goal-oriented

IV.5 Discussion & Summary

IV.6 Bibliography
Terminology

- requirements specification. A document that specifies the requirements for a system or component. Typically included are functional requirements, performance requirements, interface requirements, design requirements, and development standards. Contrast with: design description. See also: functional specification; performance specification.

[IEEE-Std-610.12-1990]

[Thayer&Dorfman1997]
A Good Requirement Specification

- Correct
- Unambiguous
- Complete
- Consistent
- Ranked for importance and/or stability
- Verifiable
- Modifiable
- Traceable

[Thayer&Dorfman1997]
[IEEE Std. 830-1993]
[IEEE Std. 830-1998]
Software Requirements Specification (1/4)

Table of Contents
1. Introduction
   1.1 Purpose
   1.2 Scope
   1.3 Definitions, acronyms, and abbreviations
   1.4 References
   1.5 Overview
2. Overall description
   2.1 Product perspective
   2.2 Product functions
   2.3 User characteristics
   2.4 Constraints
   2.5 Assumptions and dependencies
3. Specific requirements
   3.1 External interface requirements
   3.2 Functional requirements
   3.3 Performance requirements
   3.4 Design constraints
   3.5 Software system attributes
   3.6 Other requirements
Appendixes
Index

[IEEE Std. 830-1998]
Software Requirements Specification (2/4)

3.1 External interface requirements
3.1.1 User interfaces
3.1.2 Hardware interfaces
3.1.3 Software interfaces
3.1.4 Communications interfaces

- A detailed description of all inputs into and outputs from the software system.
- Complements the interface descriptions in 2.1 (not repeated information)

Content/format:
- Name of item;
- Description of purpose;
- Source of input or destination of output;
- Valid range, accuracy, and/or tolerance;
- Units of measure;
- Timing;
- Relationships to other inputs/outputs;
- Screen formats/organization;
- Window formats/organization;
- Data formats;
- Command formats;
- End messages.
Software Requirements Specification (3/4)

3.2 Functional requirements
3.2.1 Mode 1
3.2.1.1 Functional requirement 1.1
...
3.2.m Mode m
3.2.m.1 Functional requirement m.1
...
3.2.m.n Functional requirement m.n

Organized using system mode
- Some systems behave quite differently depending on the mode of operation (training, normal, or emergency)

Alternatives:
- User class
- Objects
- Feature
- Stimulus
- Response
- Functional hierarchy

[IEEE Std. 830-1998]
3.3 Performance requirements
static and the dynamic numerical requirements placed on the software or on human interaction with the software

a) The number of terminals to be supported;
b) The number of simultaneous users to be supported;
c) Amount and type of information to be handled.
d) Amount of data to be processed within certain time periods for both normal and peak workload conditions.

3.4 Design constraints
This should specify design constraints that can be imposed by other standards, hardware limitations, etc.

3.5 Software system attributes
Examples

- Reliability
- Availability
- Safety
- Security
- Maintainability
- Portability

3.6 Other requirements

[IEEE Std. 830-1998]
IV.3 Approach: SysML

IV.1 Requirements Engineering
IV.2 Requirement Specification

IV.3 Approach: SysML

IV.4 Approach: Goal-oriented
IV.5 Discussion & Summary
IV.6 Bibliography
SysML – An Upcoming Standard

System modeling language (SysML):
- defines a modeling language for systems engineering applications
- supports the specification, analysis, design, verification and validation of a range of complex systems
- intended to assist in integrating systems and software methodologies

Who?

Why Model based?
- Improved communications
- Reduced ambiguity
- Reduced errors
- More complete representation
- Enhanced knowledge capture

[SysML1.0alpha]
From Document centric to Model centric

Past

- Specifications
- Interface requirements
- System design
- Analysis & Trade-off
- Test plans

Future

Here: only Requirements!
SysML & UML 2.0

Common diagrams: Activities, Block Definitions (UML2::Classes), Internal Blocks (UML2::Composite Structures), Sequences, State Machines, Use Cases

New diagrams: Allocations, Parametric Constraints, Requirements
SysML Diagrams

(1) Requirement Diagrams
(2) Use Case Diagrams
(3) Scenarios: Sequence/Activity Diagrams
Sample Problem

The sample problem describes the development of a Hybrid Sports Utility Vehicle (SUV) system. The problem is derived from a marketing analysis which indicated the need to increase the fuel economy and “eco-friendliness” of the vehicle from its current capability without sacrificing performance. Only a small subset of the functionality and associated vehicle system requirements and design are addressed to highlight this application.
(1) Requirement Diagram

A Requirement specifies a capability or condition that a system must satisfy. A requirement may define a function that a system must perform or a performance condition that a system must fulfill. Requirements are used to establish a contract between the customer (or other stakeholder) and those responsible for designing and implementing the system.

A composition describes how a compound requirement can be decomposed into multiple sub-requirements.

A Derive relationship is a trace dependency between a derived requirement and a source requirement, where the derived requirement is generated or inferred from the source requirement.

A Satisfy relationship is dependency between a supplier requirement and a client model element that fulfills the requirement.

A test case is a behavior or operation that specifies how a requirement is verified. A test case can address one or more verification methods. A test case always returns a verdict.

A Verify relationship is a trace dependency between a supplier requirement and a client test case that determines whether a system fulfills the requirement.
Requirement Decomposition

Requirement Diagram: Top-Level User Requirements

- UR1.1: Load
  - UR1.1.1: Passengers
  - UR1.1.2: Fuel Capacity

- UR1.2: Eco-Friendliness
  - UR1.2.1: Emissions
    - UR1.2.1.1: "The car shall meet 2010 Kyoto Accord emissions standards."
    - UR1.2.1.2: "Users shall obtain fuel economy better than that provided by 95% of cars built in 2004."

- UR1.3: Performance
  - UR1.3.1: Acceleration
  - UR1.3.2: Braking
  - UR1.3.3: Power
  - UR1.3.4: Range

- UR1.4: Ergonomics

SysML1.0alpha
A Measure of Effectiveness (MoE) states an optimization condition that a system must satisfy. Whereas the requirements for a system define the domain of the solution, the solution space, the Measures of Effectiveness drive the solution to a particular region in that space. Each MoE has a weight attribute to reflect its relative importance and a score attribute to capture its value based on the alternative under investigation.

**Trade Analysis**

![Table](image)

<table>
<thead>
<tr>
<th>«effectiveness»</th>
<th>MOEName</th>
</tr>
</thead>
<tbody>
<tr>
<td>score : Real</td>
<td>weight : Real = 100</td>
</tr>
</tbody>
</table>

```plaintext
«effectiveness»
id = "MyMOEIdentifier"
text = "MOE description"
optimizationDirection = maximize
```

![Diagram](image)
Derived Requirement

[SysML1.0alpha]

Requirements Diagram: Requirement Derivations

- **Requirement:** ::HybridSUV::Performance ::FuelEconomy
  - Derived from: ・Regenerative Braking
  - Derived from: ・PowerSourceSelection

- **Requirement:** ・Regenerative Braking
  - ID: SR3.5.1
  - Source: "Trade Study 13-25"
  - Text: "The vehicle shall convert kinetic energy to electrical energy to recharge the battery and provide additional braking capability."
  - reqType: String = "Functional"
  - verifyMethod: String = "Test"
  - risk = "Medium"

- **Requirement:** ・PowerSourceSelection
  - ID: SR3.5.1
  - Source: "Trade Study 13-25"
  - Text: "If the vehicle velocity is below 10 km/h the electric motor shall be used as the sole power source. If the vehicle velocity is between 10 km/h and 50 km/h the internal combustion engine shall be used as the sole power source. If the velocity of the vehicle is greater than 50 km/h the electric motor and internal combustion engine shall both be used as power sources."
  - reqType: String = "Functional"
  - risk = "Medium"

- **Requirement:** ::HybridSUV::Eco-Friendliness ::Emissions
  - Derived from: ・Regenerative Braking

- **Requirement:** ::HybridSUV::Performance ::Range
  - Derived from: ・Regenerative Braking

- **Requirement:** ::HybridSUV::Performance ::Acceleration
  - Derived from: ・Regenerative Braking

- **Requirement:** ::HybridSUV::Performance ::Braking
  - Derived from: ・Regenerative Braking
Requirements Satisfaction

Requirement Diagram: System Requirement PowerSourceSelection Satisfaction

- «requirement» PowerSourceSelection
  - «satisfy» «usecase» ::SUV::UseCases::Accelerate
  - «satisfy» «block» ::SUV::Blocks::EngineControlUnit
Requirements Verification

Requirement Diagram: Requirement Verification

«requirement»
Regenerative Braking

«verify»
«verify»

«interaction, testCase»
::SUV::Tests::ConvertKineticToElectrical

«activity, testCase»
::SUV::Tests::ConvertKinetic
Traceability
(2) Views & Viewpoints

- A View is an abstraction of a whole system that addresses one or more concerns of the system stakeholders. A view has only one viewpoint.

- A Viewpoint specifies the purpose, stakeholders, stakeholder concerns, language selections and method selections related to a view.

- A Conform relationship is dependency between a supplier viewpoint and a client view that fulfills the requirement.

Remark: The definitions of view and viewpoint used by SysML are intended to be compatible with the IEEE 1471 Recommended Practice for Architecture Description. [IEEE-Std-1471-2000]
Examples

- **<<view>> Functional**
  - «viewpoint» Functional
  - purpose = “To define ...”
  - stakeholders = “Systems Engineers, ...”
  - concerns = “What is responsible for”
  - languages = “SysML”
  - methods = “analysis components, ...”

- **<<view>> Operational**
  - «trace»
- **<<view>> Functional**
  - «trace»
- **<<view>> Design**
  - «trace»
(2) Use Case Diagram

Aids by establishing:

- the scope and context of the system under development (:HybridSUV),
- identifying key external entities (people, external systems, etc.) that interact with the system along with the associated external interfaces, and
- providing the initial high level decomposition of behavior according to key system threads or scenarios.
Example
Refining Use Cases

- Scenarios
  - Sequence Diagrams
  - Activity Diagrams

See Chapter V
IV.4 Approach: Goal-oriented

IV.1 Requirements Engineering
IV.2 Requirement Specification
IV.3 Approach: SysML

IV.4 Approach: Goal-oriented

IV.5 Discussion & Summary
IV.6 Bibliography
Conceptualising Systems Using Goals

- We can understand an envisioned system in terms of the goals it is intended to meet.
- A goal is an expression of a state to be achieved or avoided.
- There are high-level goals and low-level goals.

Deriving goal hierarchies:
- Top-down decomposition of high-level goals.
- Elicit goals from the stakeholders; then build a hierarchy from the elicited goals.
- Infer the existence of goals.
Decomposing Goals

- High-level goals may be decomposed into lower-level goals in two ways to produce a goal hierarchy:

1. **Or Decomposition:**
   - G1 is achieved if either G1.1 or G1.2 is achieved.

   ![Or Decomposition Diagram]

2. **And Decomposition:**
   - G1 is achieved if G1.1 and G1.2 are achieved.

   ![And Decomposition Diagram]
A Goal Hierarchy

- Refine until leaf goals can be satisfied
- Multiple „implementation“ alternatives!
Conflicting Goals

- Sometimes goals in a goal hierarchy may conflict with one another. For example, in a library system, two conflicting goals may be:

Conflict Detection:
- Manual treatment does not **scale up** well
- Alternative: re-express each goal formally and then use a **theorem prover** to detect inconsistencies, i.e. conflict.

Conflict resolution:
- return to the stakeholders who “own” the conflicting goals to see whether they would be prepared to accept a compromise (**satisficing**).
- Using only the goal with the highest priority
Inferred Goals

- Minimise time to resolve user problems
- Facilitate access to problem information
- Match expert to problem
- Provide access to problem solutions

This goal has been inferred from the three lower level goals.
IV.5 Discussion & Summary

- Requirements is a **crucial element** of developing systems as about 80% of the software errors are located in the requirements.

- The **system modeling language (SysML)** provides a modeling language for systems engineering applications that supports the specification and analysis of requirements and integrates systems and software methodologies.

- **Goal-oriented requirements engineering** facilitates the detection and resolution of intrinsic conflict and facilitates the identification and exploration of alternative solutions.
IV.6 Bibliography (Additional ones)


ISBN: 0201596202


Bibliography (Additional ones)


