L3 – Documentation

DAT232/DIT285 Advanced Requirements Engineering

Eric Knauss
eric.knauss@cse.gu.se

2025-Sep-9

Outline

Housekeeping

Discussion points

- Meeting your supervisor – please make individual meetings as soon as possible (15-30min or so, this week)
- Making time for workshops
  - Those workshops benefit from you working in a group
  - We arrange several meetings with a small number of groups. All of those are there for your benefit.
  - Please prioritize these meetings over anything that is easier to reschedule
  - If absolutely necessary, reach out to other groups to switch slots
  - If nothing else helps, tell me, don't participate, and expect rework

Course representatives

Housekeeping

- A course representative collects experiences and improvement suggestions during the course and gives this feedback to the teachers
- All course members are encouraged to talk to the teachers about how the course work!
- When the course evaluation is ready, the course representatives meets with the course responsible and discusses the working report.
- Thanks to those who volunteered! Here are the course representatives (based on time of volunteering, random selection, and diversity goals):

Name	Email	Program
Murugan Raja rajeswari, Kavusalya	gusmurugka@student.gu.se	Software Engineering and Management, UGOT
Andrei-Laurențiu Balgradean	andbalg@chalmers.se	Software Engineering and Technology, CTH
Muhammad Arslan Zia	arslan.zia11@gmail.com	Game design and Technology, UGOT

First meeting: Friday, after lecture.

Feedback from Socrative

11. Please name one thing that you liked and one thing that you wished for in this lecture.

Hide Answers | Show Names | 9/92 Students Answered

- when we talked about domain level and product level elicitation, often some techniques could belong to either category, and we accept this ambiguity. Is it necessary to label things and put them in boxes, if we will anyway leave room for ambiguity? it feels like over complicating things.
- the interactiveness. its a fun wake up call to answer questions
- j
- i like the lectures themself, i like the flow i like socrative sad we didn't get to go through all of the slides (the hurry in the end)
- i like the examples given and direct approach to the issues about our project
- I liked that you explained pros and cons of different elicitation techniques, it was a very helpful refresher.
- I enjoyed the interactive aspects of the lecture
- I liked the slides, not too much or too little text. I wish the lecturer talked at a slightly faster pace.
- More examples when talking about these concepts and interactions for students participate and deepen understanding.

12. In the advanced RE course, I would like to learn more about...

Hide Answers | Show Names | 8/92 Students Answered

- unsure
- v
- how to write those documents without suffering by being tied to strict rules
- more real world examples
- –
- How RE is used in the industry, such as examples
- .
- Distinguish different concepts, ideas and techniques.

Learning Objectives

Knowledge	Skills	Judgement
K1 Identify a common RE challenge in a given software development context.	S1 Plan suitable RE practices in a team with respect to a given software development context.	J1 Assess new requirements engineering knowledge (challenge, principle, practice) and relate them to the framework in this course.
K2 Choose an appropriate RE practice in a given software development context.	S2 Effectively apply a suitable RE practice in a team in a given software development context.	J2 Suggest suitable actions to overcome a lack of requirements knowledge in a software development context.
K3 Compare suitability as well as advantages and disadvantages of given RE practices in a given software development context.	S3 Analyze the effect and quality of the outcome of a set of or individual RE practices in a given software development context.	J3 Consider inter-team, program level and social/ethical implications of a set of RE practices in a given software development context.
K4 Explain the current state of practice and research in requirements engineering.		J4 Critically assess the effectivity of a set of RE practices from the perspective of the student's master program (e.g. Software Engineering & Technology/Management, Interaction Design, Game Design, Data Science, ...)

Learning Objectives

What are key concepts of requirements documentation and specification?

Theory and Definitions

Why is requirements documentation important?

Audience and Purpose

What are the main challenges of requirements documentation?

Technical Writing for Mixed Audience, Conceptual Weaknesses

How to structure requirements documentation?

Being able to make a proposal

Outline

Things to cover from L2

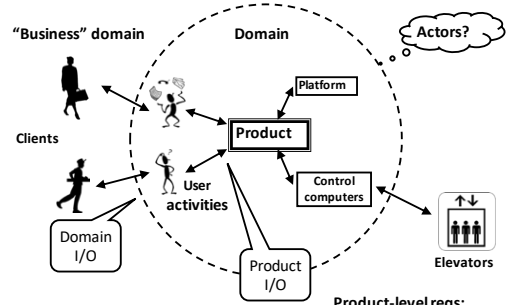
- What is E. / Why is E. difficult: Must know. (= Likely in Exam)
- Stakeholder Map and Analysis: Must know. (= Use in Project and likely in Exam).
- Overview of Elicitation Methods: Must know.
- Interviews and Strategies for Elicitation: Should know.
- Common interview mistakes: additional training if needed.
- Focus group: Must know. Read in book!
- Goal-Domain Tracing: Must know. Read in book!

Scopes

Requirements and their sources

- Business case (or: goal-level) requirements
- User (or: domain-level) requirements
- System (or: product-level) requirements
- Design-level requirements

Fig 1.5A Domain and product level



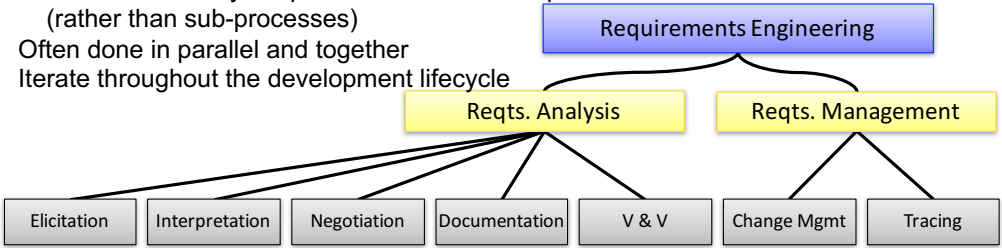
From: Soren Lauesen: Software Requirements
© Pearson / Addison-Wesley 2002

[Lauesen, 2002]

Activities

What is Requirements Engineering?

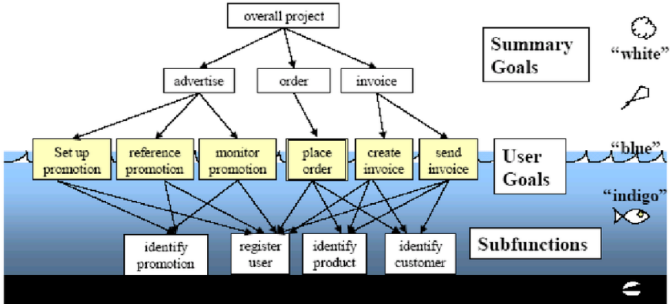
A number of loosely coupled activities and competencies
(rather than sub-processes)
Often done in parallel and together
Iterate throughout the development lifecycle



Src: DaimlerChrysler, Dagstuhl-Seminar 1998

Abstraction levels

What is Requirements Engineering?

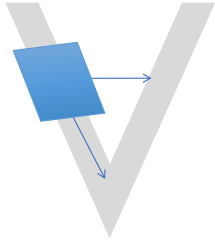


[Cockburn, 2001]

[Gorschek and Wohlin, 2006]

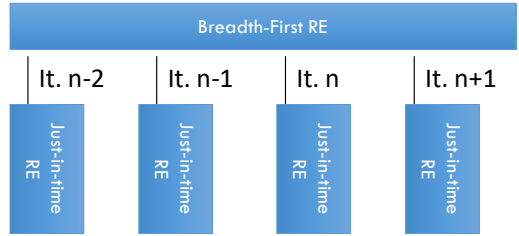
Requirements Engineering: Organizational aspects

Then



Requirements a “waterfall phase” with specialists

Now



Requirements are everybody’s responsibility

A knowledge management problem

“Individuals and interactions [...] over comprehensive documentation”

[agilemanifesto.org]

Outline

What is requirements documentation

	A requirement is [IEEE, 1990]:	
(1)	a condition or capability needed by a user to solve a problem or achieve an objective	user requirements
(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 document	system requirements
(3)	a documented representation of a condition or capability as in (1) or (2)	documentation

The role of documentation

Dan Berry[1]

Requirements Engineering is:

How to squeeze requirements out of the client's mind without damaging the client!

Elicitation is:

How to squeeze information out of the client's mind without damaging the client?

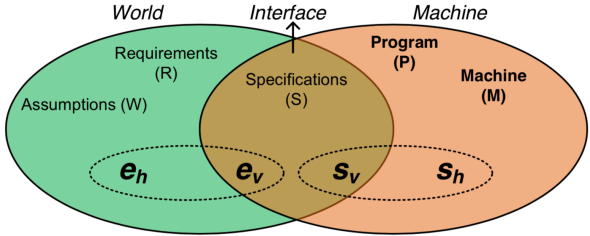
Analysis is:

How to squeeze as much additional information as possible out of what has been obtained by squeezing the client!

Tools and Environments deal with:

How to automate the storage of information before and after analytic squeezing as well as all kinds of squeezing!

[1] Quotes from https://cs.uwaterloo.ca/~dberry/COURSES/requirements.engr/lectures.pdf/iceberg.pdf



Some theory[2]

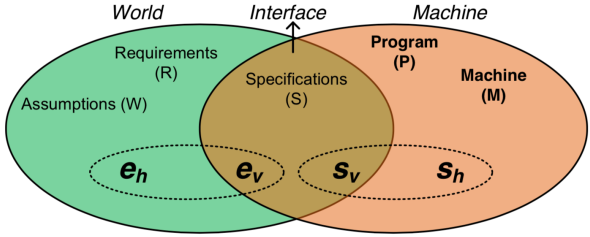
What is documentation

→ socrative.com, REQENG

The WRSPM model [Gunter et al., 2000] extends the seminal World-Machine model [Jackson, 1995] and defines requirements artifacts based on states s and events e.

Concept	*Definition*
R / W	*Requirements / Assumptions about World:* Expressed over phenomena in the real world (e_v and e_h) and those shared with the machine (s_v)
P / M	*Program / Machine (the program runs on):* Expressed over phenomena that belong to the machine (s_v and s_h) and those shared with the real world (e_v)
S	*Specification:* Only expressed over the shared phenomena at the interface (e_v and s_v).

[2] Too complicated? Read Dan Berry's excellent slides at https://www.student.cs.uwaterloo.ca/~se463/Slides/RE-referenceModel.pdf



Some theory[3]

What is documentation

The WRSPM model [Gunter et al., 2000] provides the essential relationships between the artifacts:

- *Relative consistency* helps ensure the consistency between the specification given the domain knowledge.
- *Domain adequacy* establishes the non-trivial existence of an environment in which requirements must be satisfied
- *Adequacy* establishes satisfaction of requirements by the composition of specification and assumptions

[3] Need an example?
https://www.student.cs.uwaterloo.ca/~se463/Slides/SelfHairCut.pdf

Some theory

What is documentation

What did we learn?

- Context is important!
- The distinction between requirements and specification deserves attention; Adequacy is about what the stakeholders really need vs. what we were able to specify.
- Then, in my view, adequacy relates to validating requirements: Do we have the right requirements?
- Relative consistency relates to verification: Did we specify the requirements correctly with respect to what we know?

Problems with this view:

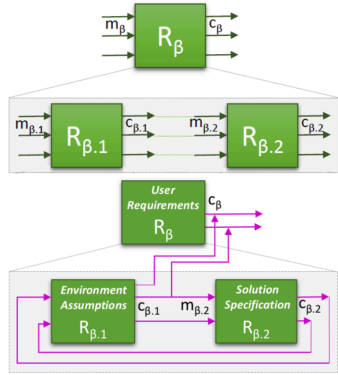
- Requirements do not always immediately come from end-users or similar stakeholder.
- Twin-Peaks model [Nuseibeh, 2001] for example describes how requirements and architecture co-evolve and influence each other.
- This implies decomposition, for which the WRSPM model does not offer direct support.

Some theory

What is documentation

This is addressed by the hierarchical reference model [Murugesan et al., 2019]:

- Allows decomposition and to check for consistency
- Provides hierarchical definitions of *consistency*, *satisfaction*, and *acceptability*
- Defines decomposition (describing logical sub-components) to architecture and explains it relation to requirements



10 min Break

Outline

Why is requirements documentation important?

- Coordinate and plan work
  - ...inform architects
  - ...inform developers
  - ...inform testers
  - ...inform line and project management (can be input to work breakdown structure and staffing decisions)
- Long-term memory: Why did we build the system that way?
- Manage liabilities
  - e.g. safety-critical systems
- Foundation for customer/supplier contract

Outline

Why is requirements documentation difficult?

- Conceptual challenges
  → Lack of conceptual framework beyond individual contexts
- Technical writing challenges (inherent to uni-directional communication)
  → no feedback, hard to ask questions,
  → hard to establish common ground

Conceptual challenges

Why is requirements documentation difficult?

Functional requirement

Two main streams of defining them:

1. What the system should do / a function that a system must be able to perform.
2. Functional (or: operational) requirements are a description of behavioral aspects of a system, i.e. specifying the inputs to the system, the outputs from the system and behavioral relationships.

Main difference: Are timing requirements functional?
→ socrative.com, REQENG, Q3

- And then: All other requirements are Non-Functional (but still must work :-)

Conceptual challenges

Why is requirements documentation difficult?

Martin Glinz most influencial paper award at RE 2017 [Glinz, 2007]

Functional vs. Non-Functional Requirements

- What vs. how the product should do (things)?

Conceptual challenges

Why is requirements documentation difficult?

Martin Glinz most influencial paper award at RE 2017 [Glinz, 2007]

Functional vs. Non-Functional Requirements

- What vs. how the product should do (things)?
- Not a good distinction.

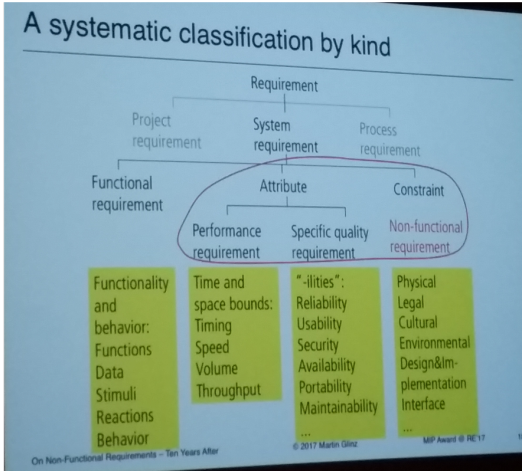
Conceptual challenges

Why is requirements documentation difficult?

Martin Glinz most influencial paper award at RE 2017 [Glinz, 2007]

Functional vs. Non-Functional Requirements

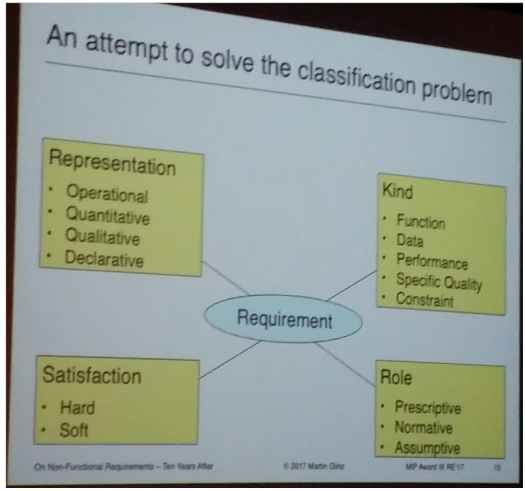
- What vs. how the product should do (things)?
- Not a good distinction.
- Here a useful classification:



Conceptual challenges

Why is requirements documentation difficult?

...and a more academic classification
[Glinz, 2007]



Conceptual challenges

Why is requirements documentation difficult?

Some definitions based on [Glinz, 2007]:

- A **concern** is a matter of interest in a system.
- The set of all requirements of a system is partitioned into **functional requirements**, **performance requirements**, **specific quality requirements**, and **constraints**.
  - A functional requirement is a requirement that pertains to a functional concern.
  - A performance requirement is a requirement that pertains to a performance concern.
  - A specific quality requirement is a requirement that pertains to a quality concern other than the quality of meeting the functional requirements.
  - A constraint is a requirement that constrains the solution space beyond what is necessary for meeting the given functional, performance, and specific quality requirements.
  - An attribute is a *performance requirement* or a *specific quality requirement*.
- A non-functional requirement is an *attribute* of or a *constraint* on a system.

10 min (Breakout) Group discussion

Give examples from your project scope: → socrative.com, REQENG, Q4-8

- a functional requirement
- a performance requirement
- a specific quality requirement
- a constraint
- a project requirement
- a process requirement

Technical writing challenges

Why is requirements documentation difficult?

Reminder: Challenges of elicitation

- Tacit knowledge (unknown knowns)
- Symmetry of ignorance (don't know what they need to know)
- Gulfs of understanding (don't know their language)

How to write down requirements in the face of these challenges?

- Invest a lot in technical writing (quality criteria on next slides)
- Rely on face-to-face communication and strong feedback cycles (agile approaches, but what about scale?)

Quality criteria

Why is requirements documentation difficult?

A good requirements specification is [Lauesen, 2002, Fig.9.1]:

Correct	Each requirement reflects a need.
Complete	All necessary requirements included.
Unambiguous	All parties agree on meaning.
Consistent	All parts match, e.g. E/R and event list.
Ranked for importance and stability	Priority and expected changes per requirement.
Modifiable	Easy to change, maintaining consistency.
Verifiable	Possible to see whether requirement is met.
Traceable	To goals/purposes, to design/code.
Additional:	
Traceable	from goals to requirements.
Understandable	by customer and developer.

Outline

How to structure

Templates provide structure and advice for content. But:

- *"**Beware of template blindness***
  Using a template easily causes template blindness: Your world view narrows down to what the template deals with.
  - *It doesn't cover everything [...]*
  - *It comprises too much [...]*
  - *It includes very demanding requirements [...]"*

[Lauesen, 2017]

How to structure

- Creating groups of five
- Provide examples of requirements documents
  - Based on the course book (Lau, [Lauesen, 2017]),
  - the IEEE Spec[4] [IEEE, 1998],
  - Volere[5], [Robertson and Robertson, 2013],
  - Experience from previous course instances and research projects
- Tasks:
  - Identify similarities
  - Identify differences
  - Which parts can be realistically covered in R1?

[4] https://web.cs.dal.ca/~hawkey/3130/srs_template-ieee.doc
[5] https://www.volere.org/templates/volere-requirements-specification-template/

The IEEE Template

How to structure

- Revision History
- Introduction
  - Purpose
  - Document Conventions
  - Intended Audience and Reading Suggestions
  - Product Scope
  - References
- Overall Description
  - Product Perspective
  - Product Functions
  - User Classes and Characteristics
  - Operating Environment
  - Design and Implementation Constraints
  - User Documentation
  - Assumptions and Dependencies
- External Interface Requirements
  - User Interfaces
  - Hardware Interfaces
  - Software Interfaces
  - Communications Interfaces
- System Features
  - System Feature 1
  - System Feature 2 (and so on)
- Other Nonfunctional Requirements
  - Performance Requirements
  - Safety Requirements
  - Security Requirements
  - Software Quality Attributes
  - Business Rules
- Other Requirements
- Appendix A: Glossary
- Appendix B: Analysis Models
- Appendix C: To Be Determined List

The Volere Template

How to structure

- Product Constraints
  - Purpose of the Product
  - Client, Customer, Stakeholders
  - Users
  - Requirements Constraints
  - Naming Conventions and Defs.
  - Relevant Facts
  - Assumptions
- Functional Requirements
  - Scope of the Product
  - Functional and Data Reqs.
- Non-Functional Requirements
  - Look and Feel Reqs.
  - Usability Reqs.
  - Performance Reqs.
  - Operational Reqs.
- Non-Functional Requirements (continued)
  - Maintainability and Portability
  - Security Reqs
  - Cultural and Political Reqs
  - Legal Requirements
- Project Issues
  - Open Issues
  - Off-the-Shelf Solutions
  - New Problems
  - Tasks
  - Cutover (Roll-out)
  - Risks
  - Costs
  - User Documentation
  - Waiting Room

Our Template

How to structure

- 1. High-Level Description
  - 1.1 Goal and scope
  - 1.2 Business Case and Stakeholder Map
  - 1.3 Core Functionality
  - 1.4 Performance Requirements, Specific Quality Requirements, Constraints
- 2. User Requirements Specification
  - 2.1 Data requirements
  - 2.2 Functional Reqs.
  - 2.3 Proposed Prioritization
  - 2.4 Detailed Performance Requirements and Specific Quality Requirements
- 3. System Requirements
  - 3.1 System requirements
  - 3.2 UI Prototype
  - 3.4 Detailed Data Requirements
  - 3.5 Acceptance Tests (Optional)

Suggestions

- Start with High-Level Description
  - We will discuss Non-Functional Requirements later in detail, still, write down what matters most
  - If performance is not a concern, or if there are no constraints, remove those words.
  - This template is simplified. Look through the others (IEEE, Volere, Lauesen) in your group and see if there is something to adopt.
- Add more detail
  - First thoughts on data requirements even in R1
  - First drafts of task descriptions (see L4) even in R1
  - Define Non-Functional Requirements in R2
  - Add Prioritization in R2

Context Diagram Revisited

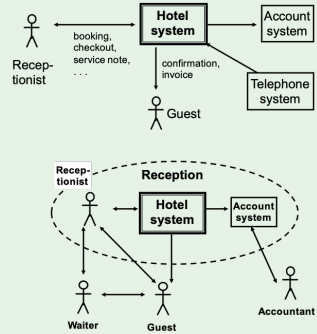
What is Requirements Engineering?

Consider

- Use Context Diagram in Section 1.1 to define Scope
- Note that while IEEE and Volere structure mainly based on requirements type, our suggestion is also to structure by requirements levels (high, user, system)

- Business case (or: goal-level) requirements
- User (or: domain-level) requirements
- System (or: product-level) requirements
- Design-level requirements

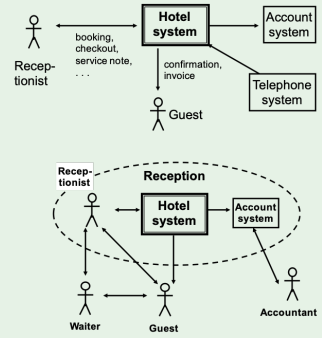
Example



On Context Diagrams

- very useful for scoping
  - consider drawing domain boundaries to distinguish goal-level/domain requirements (aka business cases) from user requirements
- useful for determining completeness
  - user level events should should be covered by functions of the system
- useful for determining technical system requirements
  - integration with existing systems
- sadly, only poorly supported by most tools, inconsistent syntax
  - provide a legend (e.g. use double line arrows for technical interfaces) and describe in text

Example



Outline

State of the art

- The question about a suitable structure, requirements information models, and tool support is at the core of our research groups' interest. The following papers indicate some of our current work and point to additional related work.
  - How requirements relate to development speed [Ågren et al., 2019, Kasauli et al., 2020a]
  - Towards identifying the minimal documentation in scaled-agile system development [Wohlrab et al., 2019b, Kasauli et al., 2020b]
  - Empower agile teams to manage system requirements with git and markdown [Knauss et al., 2018] and to manage evolving requirements models [Liebel and Knauss, 2023]
  - Consolidating requirements information models at scale, balancing diversity and alignment [Wohlrab et al., 2018, Wohlrab et al., 2019a]

Outline

Todo

- Supervised Group Work on Thursday
- L4 on Friday
- Read Lau: 6 (referring to the course book as [Lauesen, 2002] from now on) and [QR]
- Work in the project:
  - Book meeting with supervisor via email to discuss PM and steps towards R1
  - Create a context diagram
  - Create a stakeholder map
  - Plan elicitation with real stakeholders
  - Template: See suggestions in these slides

→ socrative.com, REQENG, Q9

L5 Creativity

Slot 1 (Sep-16, 15:15-17:00, Omega)

- Group 0
- Group 1
- Group 2
- Group 3
- Group 4
- Group 5
- Group 6

Slot 2 (Sep-18, 13:15-15:00, J122)

- Group 7
- Group 8
- Group 9
- Group 10
- Group 11
- Group 12
- Group 13

Slot 3 (Sep-18, 15:15-17:00, J122)

- Group 14
- Group 15
- Group 16
- Group 17
- Group 18
- Group 19
- Group 20

Slot 4 (Sep-19, 13:15-15:00, J121)

- Group 21
- Group 22
- Group 23
- Group 24
- Group 25
- Group 26
- Group 27

References I

Ågren, S. M., Knauss, E., Heldal, R., Pelliccione, P., Malmqvist, G., and Boden, J. (2019).
The impact of requirements on systems development speed: A multiple-case study in automotive.
Requirements Engineering (REEN), 24(3):315–340.

Cockburn, A. (2001).
Writing Effective Use Cases.
Addison-Wesley.

Glinz, M. (2007).
On non-functional requirements.
In *Proc. of 15th IEEE Int. RE Conf. (RE)*, pages 21–26, New Delhi, India.

Gorschek, T. and Wohlin, C. (2006).
Requirements abstraction model.
Requirements Engineering, 11:79–101.

Gunter, C. A., Gunter, E. L., Jackson, M., and Zave, P. (2000).
A reference model for requirements and specifications.
IEEE Software, 17:37–43.

IEEE (1990).
IEEE standard glossary of software engineering terminology.
IEEE Std 610.12-1990, pages 1–84.

IEEE (1998).
Ieee recommended practice for software requirements specifications.
IEEE Std 830-1998.
http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=720574.

References II


Jackson, M. (1995).
The world and the machine.
In *Proc. of 17th Int. Conf. on Software Engineering (ICSE)*, pages 283–283.


Kasauli, R., Knauss, E., Nakatumba-Nabende, J., and Kanagwa, B. (2020a).
Agile islands in a waterfall environment: Requirements engineering challenges and strategies in automotive.
In *Proceedings of International Conference on Evaluation and Assessment in Software Engineering (EASE)*, pages 31–40, Trondheim, Norway.
Download PDF at https://dl.acm.org/doi/pdf/10.1145/3383219.3383223.


Kasauli, R., Wohlrab, R., Knauss, E., Steghofer, J.-P., Horkoff, J., and Maro, S. (2020b).
Charting coordination needs in large-scale agile organizations with boundary objects and methodological islands.
In *Proceedings of International Conference on Software and System Processes (ICSSP)*, Seoul, South Korea.
Download PDF at https://arxiv.org/pdf/2005.05747.pdf.


Knauss, E., Liebel, G., Horkoff, J., Wohlrab, R., Kasauli, R., Lange, F., and Gildert, P. (2018).
T-reqs: Tool support for managing requirements in large-scale agile system development.
In *Proc. of Int. Requirements Engineering Conference (RE)*.
Tool Demo Track.


Lauesen, S. (2002).
Software Requirements.
Pearson / Addison-Wesley.
the course book (Lau).


Lauesen, S. (2017).
Guide to Requirements SL-07: Problem-oriented requirements v5.
Complements the course book (Lau) with hands on advice and a template.

References III


Liebel, G. and Knauss, E. (2023).
Aspects of modelling requirements in very-large agile systems engineering.
Systems and Software.
Download PDF at https://arxiv.org/pdf/2209.01993.


Murugesan, A., Heimdahl, M., and Rayadurgam, S. (2019).
Requirements reference models revisited: Accommodating hierarchy in system design.
In *Proc. of 27th IEEE Int. Requirements Eng. Conf. (RE)*, Jenju Island, South Korea.


Nuseibeh, B. (2001).
Weaving together requirements and architectures.
Computer, 34:115–117.


Robertson, S. and Robertson, J. (2013).
Mastering the Requirements Process: Getting Requirements Right.
Addison-Wesley Professional, 3rd edition.


Wohlrab, R., Knauss, E., and Pelliccione, P. (2019a).
Why and how to balance alignment and diversity of requirements engineering practices in automotive.
Systems and Software, 162.


Wohlrab, R., Pelliccione, P., Knauss, E., and Larsson, M. (2019b).
Boundary objects and their use in agile systems engineering organizations.
Journal of Software: Evolution and Process, 31:1–24.

References IV

Wohlrab, R., Pellicione, P., Knauss, E., and Gregory, S. (2018).
The problem of consolidating re practices at scale: An ethnographic study.
In *Proceedings of 24th Int. Working Conference on Requirements Engineering: Foundation for Requirements Engineering (REFSQ)*, Utrecht, The Netherlands.