Requirements Engineering for Software Developers

Björn Regnell

Updated: 2024-05-13
License: CC-BY-SA
https://github.com/bjornregnell/reqeng-book

Contributors:
Your name here?
contributions are welcome, contact
`bjorn.regnell@cs.lth.se`

Contents

0 Preface 4

I Domain Knowledge 5

1 Introduction 6
- 1.1 What is a requirement? 6
- 1.2 What is Requirements Engineering? 7
  - 1.2.1 Core activities of RE 7
  - 1.2.2 What is good RE? 8
  - 1.2.3 Requirements in development 8
    - Requirements and product design 8
    - Requirements and product verification 8
    - Requirements and product scoping 9
  - 1.2.4 Requirements as constraints 9
- 1.3 Specification 9
  - 1.3.1 What is a req spec? 9
  - 1.3.2 What is a good req spec? 9
  - 1.3.3 Different kinds of requirements 10
    - Functional Requirements (FR) 10
    - Quality Requirements (QR) 11
    - Requirements at different levels 12
  - 1.3.4 Feature requirements 12
- 1.4 Tools and techniques 12
  - 1.4.1 Modeling and abstraction 12
  - 1.4.2 Generic tools and techniques 12
  - 1.4.3 RE-specific language 12
    - Natural language 12
    - reqT-lang: a requirements metalanguage 12
  - 1.4.4 RE-specific tools 13

2 Context 14
- 2.1 Context of RE 14
  - 2.1.1 Business model 14
  - 2.1.2 Customer-supplier relationship 14
  - 2.1.3 Product and integration 14
  - 2.1.4 Service and delivery 14
- 2.2 Domain modeling 14
  - 2.2.1 Stakeholder modeling 14
  - 2.2.2 Goal modeling 14
  - 2.2.3 Context diagram 15

3 Elicitation 16
- 3.1 What information to elicit? 16
- 3.2 Why is elicitation so hard? 16
- 3.3 Engaging with stakeholders 16
  - 3.3.1 Surveys 16
  - 3.3.2 Interviews 16
  - 3.3.3 Case studies 17
  - 3.3.4 Creativity methods 17
- 3.4 Elicitation in product operation 17
  - 3.4.1 User engagement 17
  - 3.4.2 Telemetry and A/B-testing 17
  - 3.4.3 Business Intelligence 17

4 Prioritization 18
- 4.1 Why prioritize? 18
- 4.2 Prioritization activities 18
- 4.3 Prioritization challenges 18
- 4.4 Priority metrics 18
  - 4.4.1 What aspects to prioritize? 18
  - 4.4.2 Priority scales 19
- 4.5 Prioritization methods 19
  - 4.5.1 Categorizing 19
  - 4.5.2 Ordering 19
  - 4.5.3 Ratio-scale estimation 19
    - 100-dollar-test 19
    - Top-ten (or Top-n) 19
    - Analytical Hierarchy Process (AHP) 19

II Functionality 20

5 Data 21
- 5.1 What is Data? 21
- 5.2 Data dictionaries 21
- 5.3 Data views 21
- 5.4 Data diagrams 21
  - 5.4.1 Entity-Relationship diagrams 21
  - 5.4.2 Class diagrams 21
- 5.5 Data format specification 21
  - 5.5.1 Regular expressions (regexp) 22
  - 5.5.2 Protocol buffers (protobuf) 22

6 Logic 23
- 6.1 User-system work split 23
- 6.2 Contextual usage modeling 23
  - 6.2.1 User stories 23
  - 6.2.2 Use cases and tasks 23
  - 6.2.3 Narratives and personas 23
- 6.3 System behavior modeling 23
  - 6.3.1 State diagram 23
  - 6.3.2 Interaction diagrams 24
  - 6.3.3 Data-flow diagrams 24

7 Prototyping 25
- 7.1 Why prototype? 25
- 7.2 Prototype scoping 25
- 7.3 Prototype media 25
- 7.4 Prototype usage 26
- 7.5 Exploration strategy 26

8 Delegated Requirements 27
- 8.1 Standards as requirements 27
- 8.2 Regulatory requirements 27
  - 8.2.1 General Data Protection Regulation 27
  - 8.2.2 Artificial Intelligence Act 27
- 8.3 Test cases as requirements 27
- 8.4 Development process requirements 27

III Quality 28

9 Quality Requirements 29
- 9.1 What is product quality? 29
- 9.2 Product-level qualities 29
  - 9.2.1 Accuracy 29
  - 9.2.2 Capacity 29
  - 9.2.3 Performance 29
  - 9.2.4 Reliability 29
  - 9.2.5 Maintainability 29
- 9.3 Domain-level qualities 29
  - 9.3.1 Usability and UX 29
  - 9.3.2 Security and safety 29
- 9.4 Quality road-mapping with QUPER 30

10 Requirements Validation 31
- 10.1 Specification quality aspects 31
- 10.2 Requirements inspections 31
- 10.3 Dynamic validation 31
  - 10.3.1 Usability testing 31
  - 10.3.2 Simulation 31

11 Product scoping 32
- 11.1 Release themes and epics 32
- 11.2 Requirements status ladder 32
- 11.3 Release planning 32

12 Product verification 33
- 12.1 Unit testing 33
- 12.2 System testing 33
- 12.3 Acceptance testing 33
- 12.4 Regression testing 33

IV Special Contexts 34

13 Special Process Modes 35
13.1 Agile RE 35
13.2 Open Source RE 35
13.3 Continuous RE 35
13.3.1 CI/CD 35
13.3.2 DevOps 35
13.4 Contract-based RE 35
13.4.1 Subcontracting and integration . . 35
13.4.2 Public procurement 35
13.5 RE in product-line engineering 35
13.5.1 Product-lines and product families 35
13.5.2 Variability modeling 35

14 Special Product Types 36
14.1 RE for High-Assurance Systems 36
14.1.1 Information security audit 36
14.2 RE for Artificial Intelligens 36
14.3 RE for Embedded Systems 36
14.3.1 Hardware-Software co-design . . 36

V Appendices 37

A Getting Started with `reqT` 38

B Tool Lab: Prioritization 39

C Tool Lab: Release Planning 40

D Course Project 41
D.1 Literature Studies 41
D.2 System Ideas 41
D.3 Course Project Outline 41
D.3.1 Management roles 41
D.3.2 Release 1: Scoped 41
D.3.3 Release 2: Specified 41
D.3.4 Release 3: Validated 41
D.3.5 Assessment matrix 42

E Specification Templates 43
E.1 How to structure a linear document? . 43
E.2 IEEE 830 Specification template . . . 43

F Learning from RE Disasters 45
F.1 Swedish National Police System 45
F.2 Swedish Regional Health Care System 45

1. Introduction

Requirements Engineering (RE) is a sub-discipline of Software Engineering (SE) that is focused on the *potential features* of future software-intensive systems and their *context*, which includes users and surrounding systems.

The requirements engineering process involves activities such as eliciting, specifying, validating, and selecting requirements. Requirements engineering is both a research discipline and an engineering practice, forming the foundation for our human ability to create useful software with high quality. When you work with anything related to the decision-making process of software development and its underlying intentions you are doing requirements engineering.

1.1 What is a requirement?

The term ’requirement’ often means **something needed or wanted**. It can also refer to a documented *representation* of something needed or wanted.

The term ’requirement’ is a bit tricky as it is overloaded with several meanings. It is sometimes used as a very abstract term denoting *any* type of information entity relevant to the intentions behind system development. Sometimes we mean an optional wish or nice-to-have feature, sometimes we mean a mandatory feature without which the system would be pointless. Sometimes it refers to something explicitly decided and committed for delivery. Sometimes we mean the actual underlying need of a user that maybe still elusive and not explicitly stated, while we also may mean the carefully documented *representation* of that underlying need.

A documented representation of a requirement may very well unintendedly differ from the underlying actual need or wish, as our specification may be far from perfect. Hence, our views of an ”actual” requirement and our views of its representation may be far from similar, if the representation is bad. It all depends on how humans interpret the representation.

Also, representations of requirements come in many different kinds and shapes at different levels of abstraction and detail, and hence, it is wise to be clear and specific about what kind of requirement we mean, to avoid that our usage of the term ’requirement’ is misinterpreted.

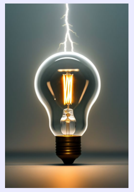
The subsequent sections aims to provide a concise overview of the set of terms that you can use when you think and communicate about different kinds of requirements in different requirements engineering contexts. This terminology is the starting-point for learning more about requirements

What is Requirements Engineering (RE)?

- RE is focused on the
  - **features** of software systems
  - **system context**, including users and connected systems
  - **development context**, including stakeholders’ intentions
- The RE process involves

• knowledge-building	research
• consensus-building	agree
• decision-making	choose
• innovation	generate ideas
• communication	be pedagogical

What is a requirement?



- A simple definition:
  - Something **needed** or **wanted**.
  - A documented **representation** of something needed or wanted.
- Are we representing what is **actually** needed or wanted?
- *’Requirement’* can in practice mean many different things: must, option, idea, innovation, intent, rationale, function, quality, design, feature, decision, constraint, ...
- The most **general** meaning: *any* kind of **entity** that *could* be included in a req spec

In Swedish:

requirements engineering	kravhantering, kravteknik, kravanalys
requirement	krav, önskemål
representation	representation, avbildning
stakeholder	intressent, aktör, sakägare