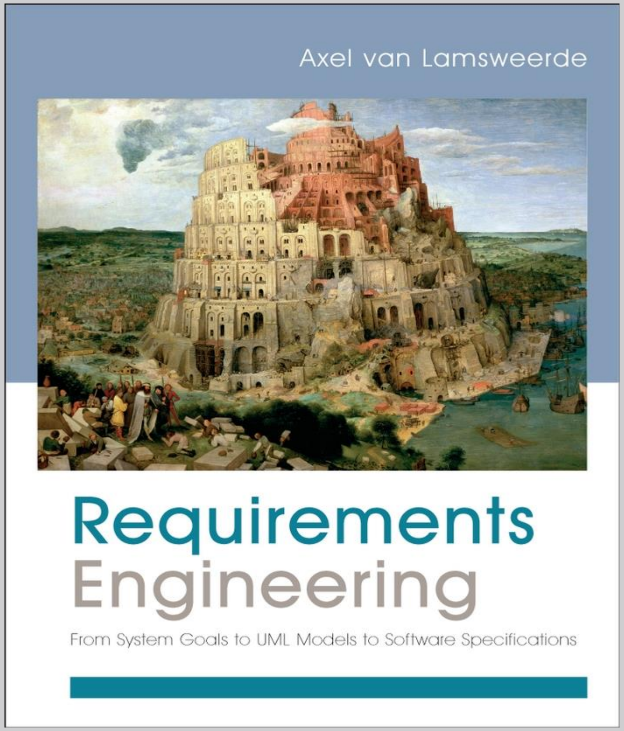
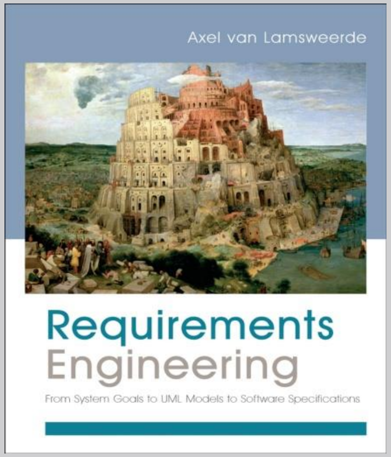
Requirements Engineering

From System Goals
to UML Models
to Software Specifications



Axel Van Lamsweerde

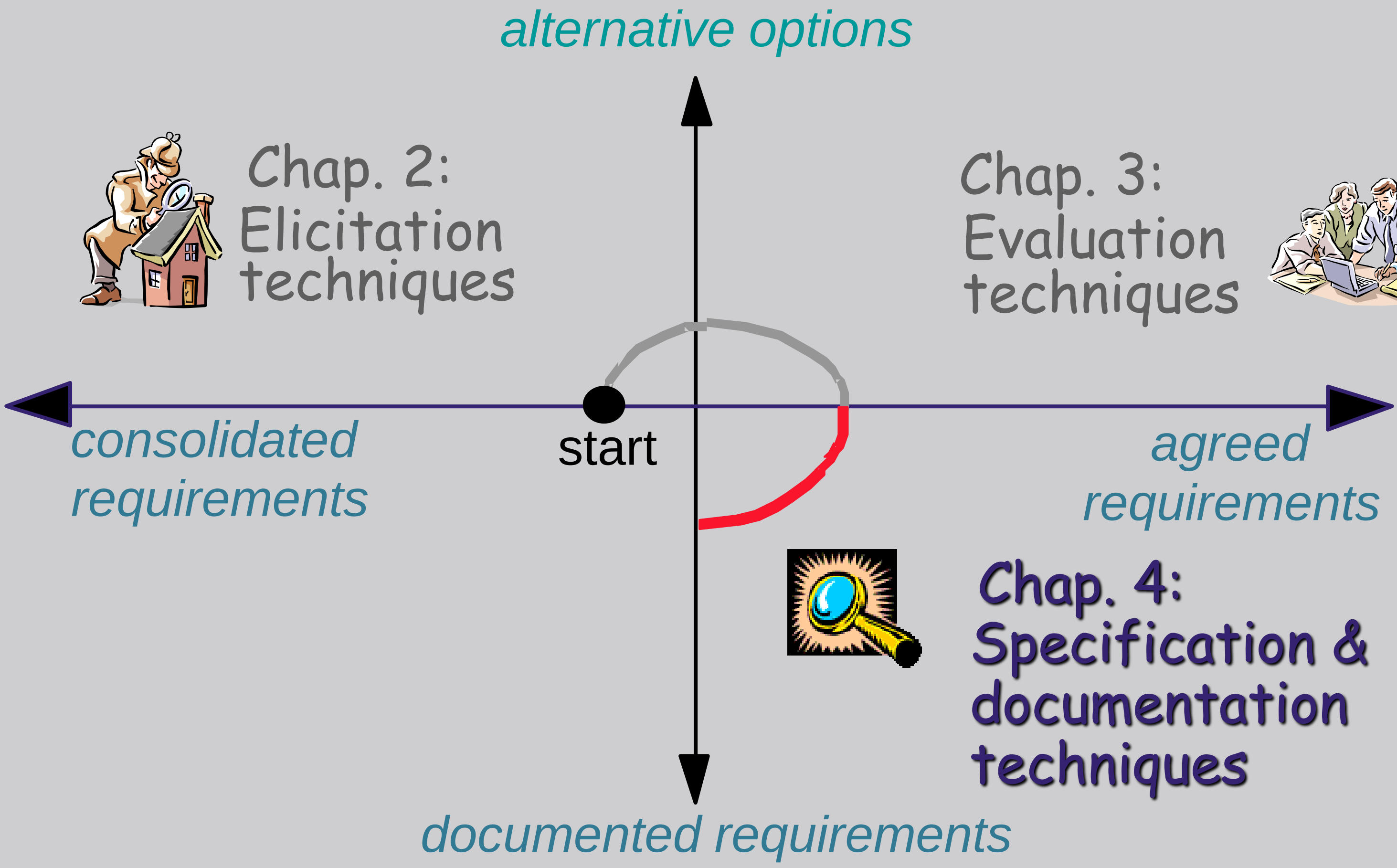


Fundamentals of RE

Chapter 4

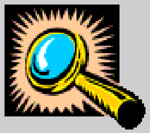
Requirements Specification & Documentation

Chap.1: RE products and processes

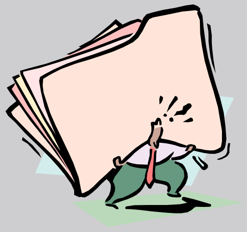


Chap. 2: Elicitation techniques

Chap. 3: Evaluation techniques



Chap. 4: Specification & documentation techniques



Specification & documentation:
as introduced in Chapter 1 ...

- Precise definition of all features of the agreed system
  - Objectives, concepts, relevant domain properties, system/software requirements, assumptions, responsibilities
  - Rationale for options taken, satisfaction arguments
  - Likely system evolutions & variants
- Organization of these in a coherent structure
- Documentation in a form understandable by all parties
  - Often in annex: costs, workplan, delivery schedules

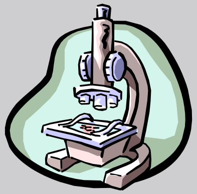
Resulting product: ***Requirements Document*** *(RD)*

Requirements specification & documentation: outline

- Free documentation in unrestricted natural language
- Disciplined documentation in structured natural language
  - Local rules on writing statements
  - Global rules on organizing the Requirements Document
- Use of diagrammatic notations
  - System scope: context, problem, frame diagrams
  - Conceptual structures: entity-relationship diagrams
  - Activities and data: SADT diagrams
  - Information flows: dataflow diagrams
  - System operations: use case diagrams
  - Interaction scenarios: event trace diagrams
  - System behaviors: state machine diagrams
  - Stimuli and responses: R-net diagrams
  - Integrating multiple system views, multi-view spec in UML

Requirements specification & documentation: outline (2)

- Formal specification
  - Logic as a basis for formalizing statements
  - History-based specification
  - State-based specification
  - Event-based specification
  - Algebraic specification



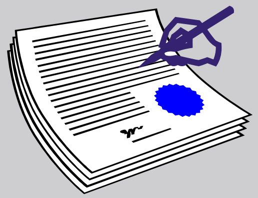
Free documentation in unrestricted natural language

- Unconstrained prose writing in natural language (NL) ...
  - ☺ Unlimited expressiveness, communicability, no training needed
  - ☹ Prone to many of the spec errors & flaws (cf. Chap.1)
- In particular, **ambiguities** are inherent to NL; can be harmful

  "Full braking shall be activated by any train that receives an outdated acceleration command **or** that enters a station block at speed higher than *X* m.p.h. **and for which** the preceding train is closer than *Y* yards."
- Frequent confusions among logical connectives in NL
  - e.g. case analysis:

    If Case1 **then** <Statement1>
    or **if** Case2 **then** <Statement2>   (amounts to **true**!)

    vs. **If** Case1 **then** <Statement1>
    and **if** Case2 **then** <Statement2>



Disciplined documentation in structured NL: local rules on writing statements

- Use **stylistic rules** for good NL spec, e.g.
  - Identify who will read this; write accordingly
  - Say what you are going to do before doing it
  - Motivate first, summarize after
  - Make sure every concept is defined before use
  - Keep asking yourself: *"Is this comprehensible? Is this enough? Is this relevant?"*
  - Never more than one req, assumption, or dom prop in a single sentence. Keep sentences short.
  - Use *"shall"* for mandatory prescriptive statements, *"should"* for desirable ones.
  - Introduce figures to provide visual overviews and emphasize key points.
  - Use suggestive examples to clarify abstract statements
  - Supply diagrams for complex relationships among items

(More in the book)

Disciplined documentation in structured NL: local rules on writing statements (2)

- Use **decision tables** for complex combinations of conditions

input **if**-conditions

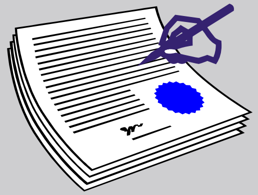
binary filling with truth values

Train receives outdated acceleration command	T	T	T	T	F	F	F	F
Train enters station block at speed $\geq X$ mph	T	T	F	F	T	T	F	F
Preceding train is closer than Y yards	T	F	T	F	T	F	T	F
Full braking activated	X		X		X			
Alarm generated to station computer	X	X	X	X				

output **then**-conditions

one case = AND-combination

- Systematic, simple, additional benefits ...
  - Completeness check: 2^N columns required for full table
  - Table reduction: drop impossible cases in view of dom props; merge 2 columns differing only by single "T", "F" => "-"
  - Test cases for free (cause-effect coverage)



Disciplined documentation in structured NL: local rules on writing statements (3)

- Use standardized **statement templates**

Identifier --suggestive; hierarchical if compound statement

Category --functional or quality req, assumption, domain property, definition, scenario example, ...

Specification --statement formulation according to stylistic rules

Fit criterion --for measurability (see next slide)

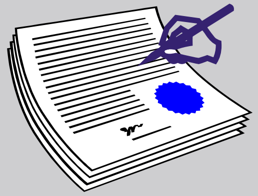
Source --for traceability to elicitation sources

Rationale --for better understanding & traceability

Interaction --contribution to, conflict with other statements

Priority level --for comparison & prioritization

Stability, **Commonality** levels --for change management

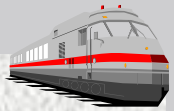


Fit criteria make statements measurable

- Complement statements by quantifying the extent to which they must be satisfied [Robertson, 1999]
- Especially important for measurability of NFRs

Spec: The scheduled meeting dates shall be convenient to participants

Fit criterion: *Scheduled dates should fit the diary constraints of at least 90% of invited participants in at least 80% of cases*



Spec: Info displays inside trains shall be informative & understandable

Fit criterion: *A survey after 3 months of use should reveal that at least 75% of travelers found in-train info displays helpful for finding their connection*

Disciplined documentation in structured NL: global rules on organizing the RD

- **Grouping** rules: Put in same section all items related to common factor ...
  - system objective
  - system component
  - task
  - conceptual object
  - software feature
  - ...
- Global **templates** for standardizing the RD structure
  - domain-specific, organization-specific, company-specific

IEEE Std-830 template for organizing the RD

1. Introduction
   - 1.1 RD purpose
   - 1.2 Product scope
   - 1.3 Definitions, acronyms, abbreviations
   - 1.4 References
   - 1.5 Overview
2. General Description
   - 2.1 Product perspective
   - 2.2 Product functions
   - 2.3 User characteristics
   - 2.4 General constraints
   - 2.5 Assumptions & Dependencies
   - 2.6 Apportioning of requirements
3. Specific Requirements

domain, scope, purpose of system-to-be

glossary of terms

elicitation sources

sw-environment boundary: interfaces with users, devices, other sw

functionalities of software-to-be

assumptions about users

development constraints (hw limitations, implem platform, ...)

environment assumptions (subject to change)

optional, deferable reqs

IEEE Std-830 template for organizing the RD (2)

3. Specific Requirements ··· *alternative templates for specific types of system*

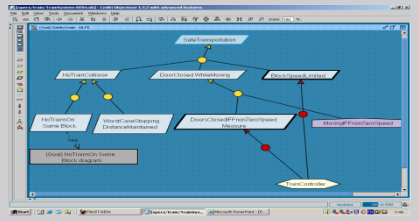
- 3.1 Functional requirements
- 3.2 External interface reqs ··· NFRs: interoperability
- 3.3 Performance reqs ··· NFRs: time/space performance
- 3.4 Design constraints ··· NFRs: development reqs
- 3.5 Software quality attributes ··· NFRs: quality reqs
- 3.6 Other requirements ··· NFRs: security, reliability, maintainability

Appendices

Index

- Variant: VOLERE template [Robertson, 1999]
  - explicit sections for domain properties, costs, risks, development workplan, ...

Use of diagrammatic notations



- To complement or replace NL prose
- Dedicated to **specific aspects** of the system (as-is or to-be)
- Graphical: to ease communication, provide overview
- Semi-formal ...
  - Declaration of items in formal language (syntax, semantics)

    => surface checks on RD items, machine-processable
  - Informal spec of item properties in NL
- **This chapter**: typical sample of frequently used diagrams, showing complementarities
- **Part 2**: in-depth study + systematic method for building complex models using integrated set of diagrams

Requirements specification & documentation (1) : summary

- Free documentation in unrestricted NL is subject to errors & flaws
- Disciplined documentation in structured NL is always necessary
  - Local rules on statements: stylistic rules, decision tables, statement templates
  - Global rules on RD organization: grouping rules, structure templates
- Diagrams for graphical, semi-formal spec of complementary aspects
  - **System scope**: context, problem, frame diagrams
  - **Conceptual structures**: entity-relationship diagrams
  - **Activities and data**: SADT diagrams
  - **Information flows**: dataflow diagrams
  - **System operations**: use case diagrams
  - **Interaction scenarios**: event trace diagrams
  - **System behaviors**: state machine diagrams
  - **Stimuli and responses**: R-net diagrams
  - Integrating **multiple views**, multi-view spec in UML