Part I: The Fundamentals

Part II: Requirements Engineering Practices

Part III: Enablers and Stumbling Blocks

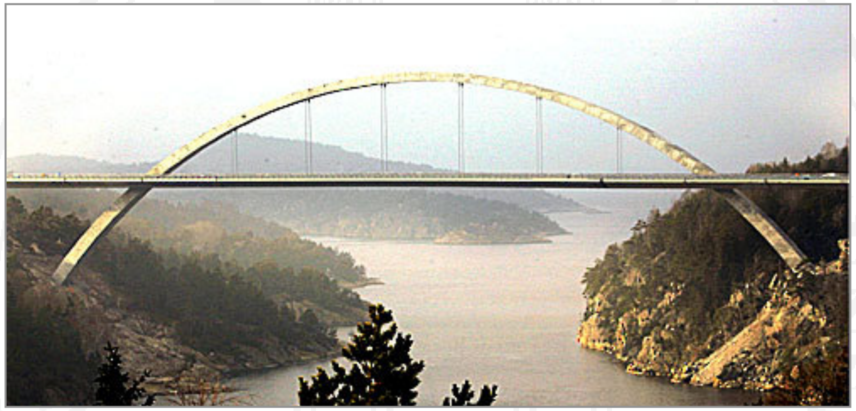
Conclusions

References

5 Documenting requirements

Bridging the gap:

Stakeholders



System builders

Photo © Lise Aserud / DPA

The need:

- Communicating requirements
- Having a basis for contracts and acceptance decisions

The means: Documented requirements

5.1 Classic requirements specifications

Full-fledged requirements specifications are typically needed

- When customers want contractually fixed requirements, costs and deadlines
- When systems are built by an external contractor based on a set of given requirements (tendering, outsourcing)
- In regulated environments where regulators check compliance of developed systems to their requirements

Document types

[ISO/IEC/IEEE 2011]

- Stakeholder requirements specification (also called customer requirements specification)
  What the stakeholders want (independent of any system providing it)
- System requirements specification
  The system or product to be developed and its context
- Software requirements specification
  If the system is a pure software system
- Business requirements specification
  High-level specification of business needs or goals

Stakeholder requirements specification

- Written when stakeholder needs shall be documented before any system development considerations are made
- Typically written by domain experts on the customer side (maybe with help of RE consultants)
- If a stakeholder requirements specification is written, it precedes and informs system or software requirements specifications

System/software requirements specification

- The classic form of a requirements specification
- No methodological difference between system requirements specification and software requirements specification
- Typically written by requirements engineers on the supplier side

Supplementary artifacts

Contain requirements-related information

May be produced in addition to a requirements specification

Typical artifacts of this kind are:

❍ Prototypes

❍ User interface mock-ups

❍ Sketches / informal drawings

❍ Glossary (if not part of the requirements specification)

5.2 Requirements in agile development

No classic requirements specification document (unless mandated by regulators)

Various artifacts / work products that ...

- ... specify requirements: vision, stories, epics, use cases,...
- ... have requirements-related content: Prototypes, mock-ups, storyboards, roadmap, early product versions (e.g., MVP – minimum viable product)

Value-driven creation of artifacts

Agile requirements specification artifacts

- Requirements primarily captured as a collection of user stories, organized in a product backlog
- A system vision provides an abstract overview of the system to be developed
- On an intermediate level of abstraction, epics and features can serve to group user stories
- Stories may be sub-divided into tasks
- Use cases/scenarios and other models may be used to provide structure and context
- Agile work products may have requirements-related contents

5.3 Aspects to be documented

Independently of any language, method, and artifact, four aspects need to be documented:

- ❍ Functionality
  - Data: Usage and structure
  - Functions: Results, preconditions, processing
  - Behavior: Dynamic system behavior as observable by users
  - Both normal and abnormal cases must be specified

Aspects to be documented – 2

- Performance
  - Data volume
  - Reaction time
  - Processing speed
  - Specify measurable values if possible
  - Specify more than just average values

- Specific qualities
  - “-ilities” such as
    - Usability
    - Reliability
    - etc.

Aspects to be documented – 3

- Constraints

  Restrictions that must be obeyed / satisfied

  - Technical: given interfaces or protocols, etc.
  - Legal: laws, standards, regulations
  - Cultural
  - Environmental
  - Physical
  - Solutions / restrictions demanded by important stakeholders

5.4 How to document

Sample standards for classic requirements documents

IEEE Std 830-1998 (outdated, but still in use)

- Three parts
- System requirements only
- Representation of specific requirements tailorable

VOLERE

- 27 chapters
- System and project requirements

Enterprise-specific standards

- Imposed by customer or given by supplier

IEEE Std 830-1998

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

Appendixes

Index

Variants:
Organize by
- Mode
- User class
- Object
- Feature
- Stimulus
- Function

VOLERE

[Robertson and Robertson 2006]
[www.volere.co.uk]

Project Drivers

1. The Purpose of the Project
2. Client, Customer & other Stakeholders
3. Users of the Product

Project Constraints

4. Mandated Constraints
5. Naming Conventions and Definitions
6. Relevant Facts and Assumptions

Functional Requirements

7. The Scope of the Work
8. The Scope of the Product
9. Functional and Data Requirements

Non-Functional Requirements

10. Look and Feel Requirements
11. Usability and Humanity Requirements
12. Performance Requirements
13. Operational Requirements
14. Maintainability and Support Requirements
15. Security Requirements
16. Cultural and Political Requirements
17. Legal Requirements

Project Issues

18. Open Issues
19. Off-the-Shelf Solutions
20. New Problems
22. Tasks
22. Cutover
23. Risks
24. Costs
25. User Documentation and Training
26. Waiting Room
27. Ideas for Solutions

Guidelines for agile requirements

- ❍ Standard template for writing user stories (cf. Chapter 8)
- ❍ Organizing stories in a product backlog
- ❍ Artifact / work product structures provided by textbooks

[Leffingwell 2011]

General guideline: do things only if they add value

How to document – language options

Informally

❍ Natural language (narrative text)

Semi-formally

❍ Structural models

❍ Interaction models

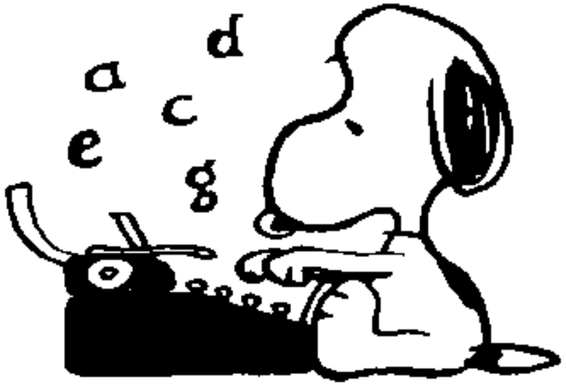
} Typically as diagrams which are enriched with natural langue texts

Formally

❍ Formal models, typically based on mathematical logic and set theory

General rules for requirements documentation

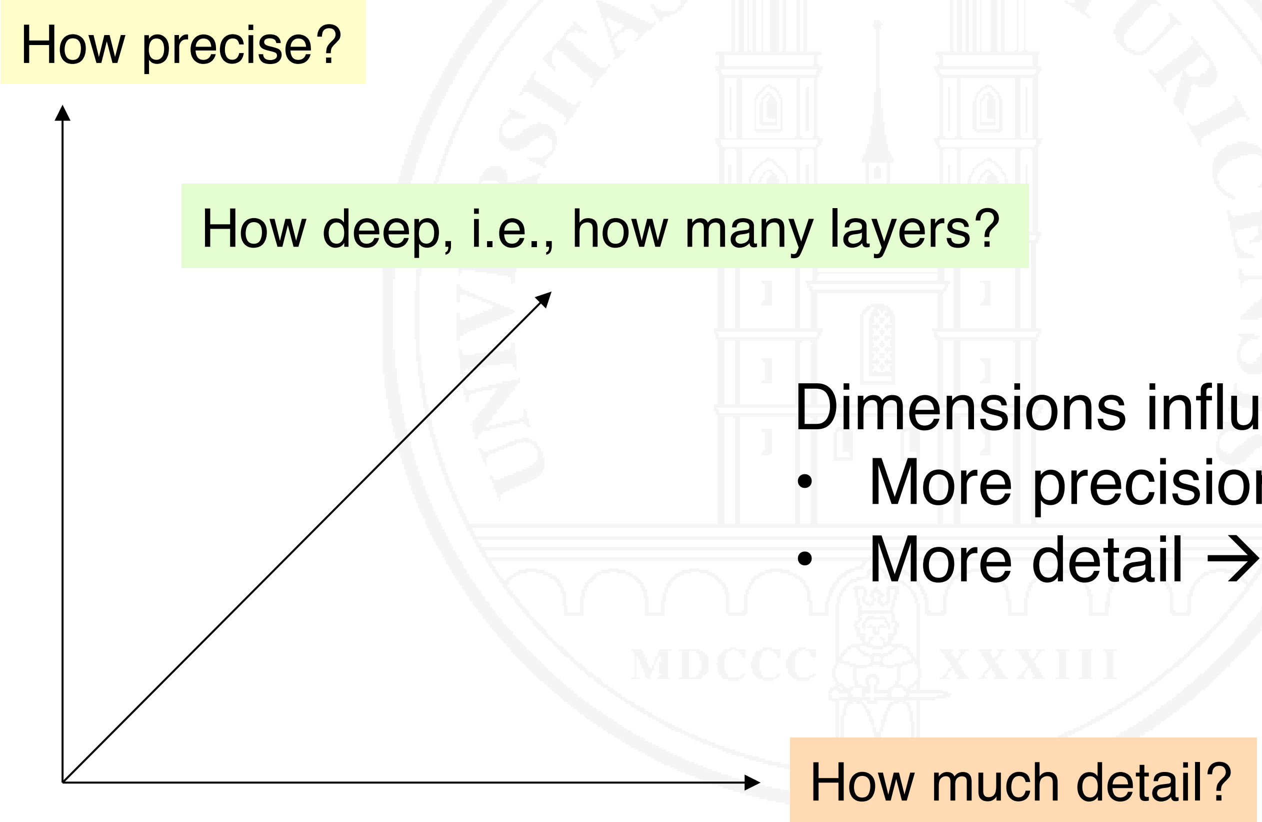
- Specify requirements as small, identifiable units whenever possible
- Record metadata such as source, author, date, status
- Use structure templates
- Adapt the degree of detail to the risk associated with a requirement
- Specify normal and exceptional cases
- Don’t forget quality requirements and constraints



© UFS, Inc.

Precision – Detail – Depth

Three dimensions:



How precise?

How deep, i.e., how many layers?

How much detail?

Dimensions influence each other:

- More precision → more detail
- More detail → more depth

Precision: reduce ambiguity

Restrict your language

Use a glossary

Define acceptance test cases

Quantify where appropriate

Formalize

Meine Liebe,
Du fehlst mir morgens, mittags und abends.
5-8
DAS IST ZU UN-GENAU...
WENN DU EINEM MÄDCHEN SCHREIBST, MUSST DU DICH PRÄZISE AUSDRÜCKEN.
Du fehlst mir um 8 Uhr 15, 11 Uhr 45 und 21 Uhr 36...
SCHULZ
© 1985 United Feature Syndicate, Inc.

Snoopy quantifies ... unfortunately, I have it only in German

Detail

What's better?

"The participant entry form has fields for name, first name, sex, ..."

"The participant entry form has the following fields (in this order): Name (40 characters, required), First Name (40 characters, required), Sex (two radio buttons labeled male and female, selections exclude each other, no default, required),..."

It depends.

- Degree of implicit shared understanding of problem
- Degree of freedom left to designers and programmers
- Cost vs. value of detailed specification
- The risk you are willing to take

Depth

The more precise, the more information is needed

➜ Preserve readability with a hierarchical structure

"…

4.3 Administration of participants

4.3.1 Entering a new participant

4.3.1.1 New participant entry form

4.3.1.2 New participant confirmation

4.3.2 Updating a participant record

…"

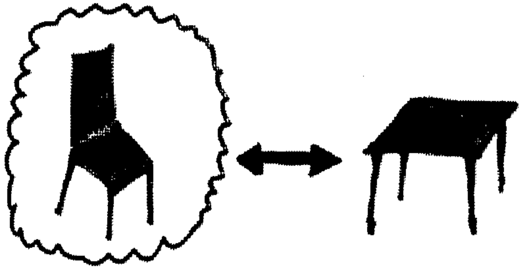
5.5 Quality of documented requirements

Two aspects of requirements quality

- Quality of individual requirements
- Quality of requirements specification documents

Hint: Don’t confuse quality of requirements with quality requirements!

Quality of individual requirements



For individual requirements, strive for requirements that are...

● Adequate	True and agreed stakeholder needs
● Necessary	Part of the relevant system scope
● Unambiguous	True shared understanding
● Consistent	No contradictions
● Complete	No missing parts
● Understandable	Prerequisite for shared understanding
● Verifiable	Conformance of implementation can be checked
● Feasible	Non-feasible requirements are a waste of effort
● Traceable	Linked to other requirements-related items

Quality of requirements artifacts

When creating a requirements specification, strive for a document that is

● Consistent	No contradictions
● Unambiguous	True shared understanding
● Structured	Improves readability of artifact
● Modifiable & extensible	Because change will happen
● Traceable	Linked to related artifacts
● Complete	Contains all relevant requirements
● Conformant	Conforms to prescribed artifact structure, format or style

Quality criteria are in the eye of the beholder

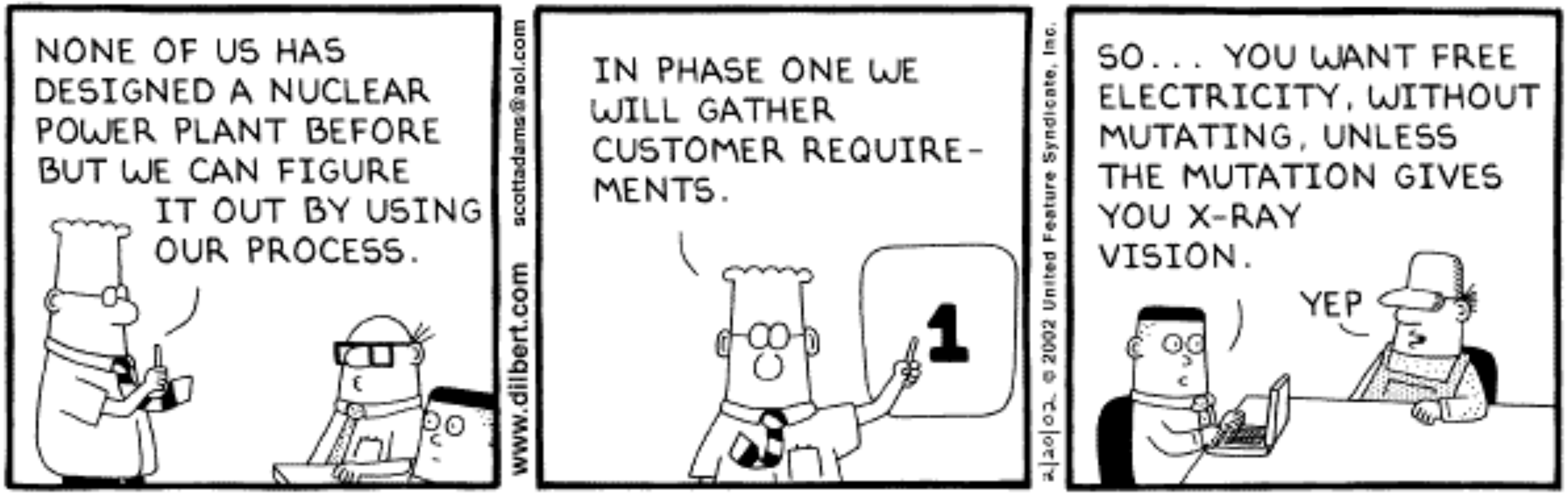
- No general consensus
- Different, overlapping sets of quality criteria used in
  - this course
  - RE textbooks
  - RE standards (e.g., ISO/IEC/IEEE 2011)
  - Quasi-standards such as the IREB Certified Professional for Requirements Engineering (see http://www.ireb.org)

Not all qualities are equally important

- Adequacy and understandability are key
- Verifiability and Consistency are very important
- Achieving total completeness and unambiguity is neither possible nor economically feasible in most cases
- The importance of feasibility, traceability, conformance, etc. of requirements depends on the concrete project/situation

☞ Strive for value, not for blind satisfaction of requirements quality criteria!

6 Requirements Engineering processes


NONE OF US HAS DESIGNED A NUCLEAR POWER PLANT BEFORE BUT WE CAN FIGURE IT OUT BY USING OUR PROCESS.
scottadams@aol.com
www.dilbert.com
IN PHASE ONE WE WILL GATHER CUSTOMER REQUIREMENTS.
1
2/20/02 © 2002 United Feature Syndicate, Inc.
SO... YOU WANT FREE ELECTRICITY, WITHOUT MUTATING, UNLESS THE MUTATION GIVES YOU X-RAY VISION.
YEP

The principal tasks

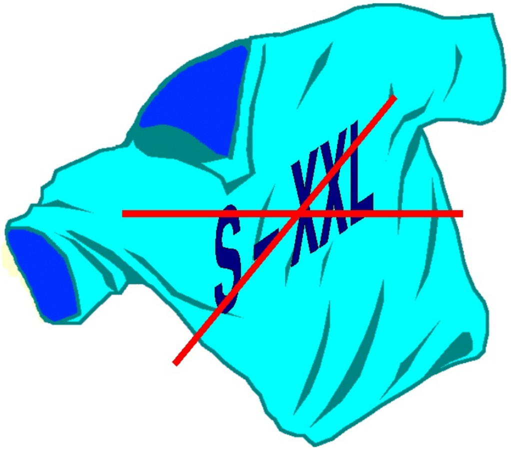
Requirements Specification

- Elicitation
- Analysis
- Documentation
- Validation

Requirements Management

- Identification and metadata
- Requirements prioritization
- Change and release management
- Traceability

No ‘one size fits all’ process

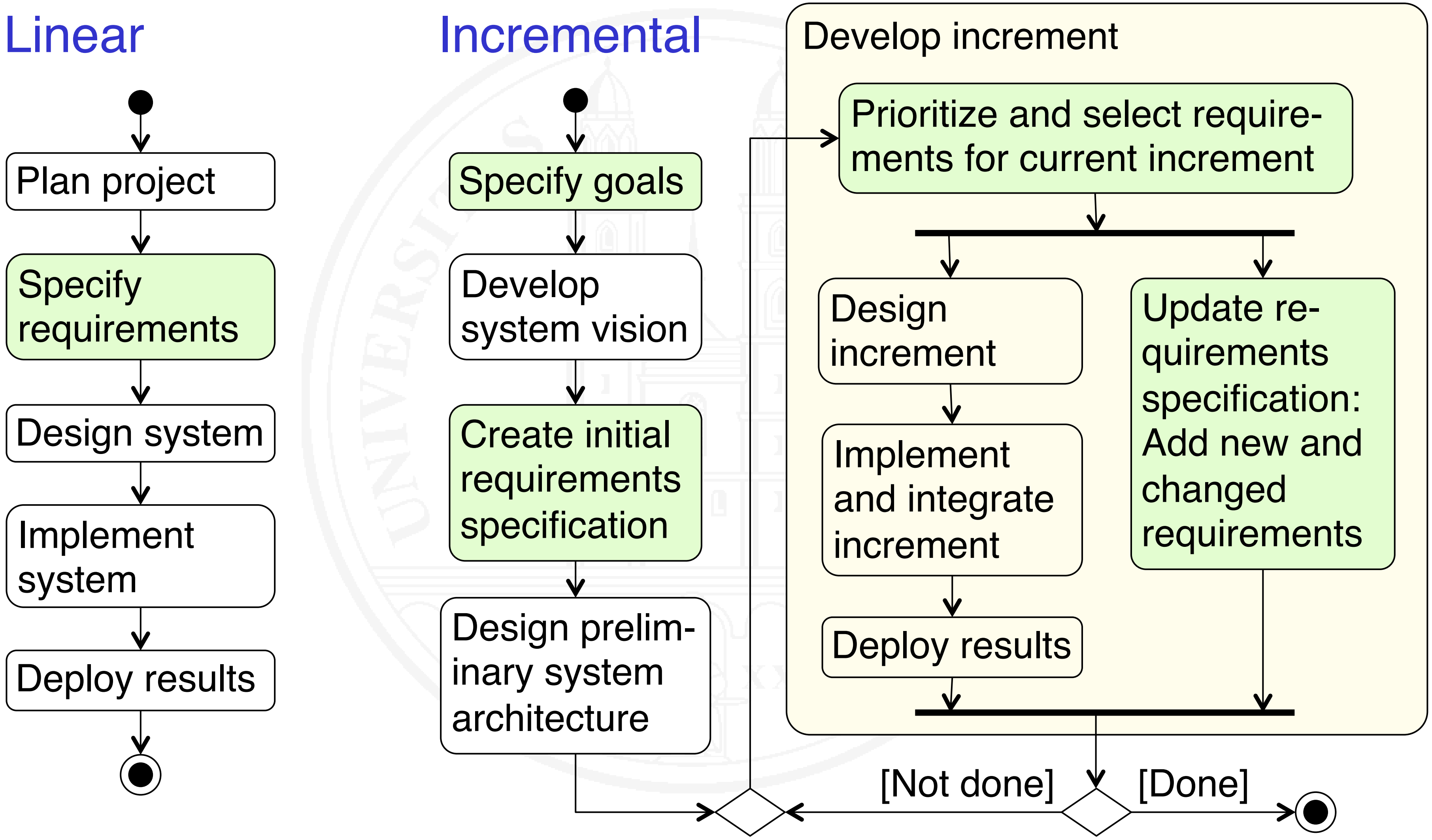


Some determining factors

- The embedding process: linear or incremental?
- Contract (prescriptive) or collaboration (explorative)?
- Can you talk with your stakeholders?
- Customer order or development for a market?
- Using COTS?

➪ Tailor the process from some principal configuration options and a rich set of RE practices

Linear vs. incremental processes


Linear
Plan project
Specify requirements
Design system
Implement system
Deploy results
Incremental
Specify goals
Develop system vision
Create initial requirements specification
Design preliminary system architecture
Develop increment
Prioritize and select requirements for current increment
Design increment
Implement and integrate increment
Deploy results
Update requirements specification: Add new and changed requirements
[Not done]
[Done]

Linear vs. incremental processes – 2

Decision criteria

- Linear
  - Clear, stable, a priori known requirements
  - Low risk (of developing the wrong system)
  - Relatively short duration of project
  - Complex requirements change process is acceptable
- Incremental
  - Evolving requirements
  - High risk (of developing the wrong system)
  - Long duration of project
  - Ability to change requirements easily is important

Prescriptive – Explorative – COTS-driven

Prescriptive process

- Requirements specification is a contract: All requirements are binding and must be implemented
- Functionality determines cost and deadlines
- Needed when tendering design and implementation
- Development of specified system may be outsourced
- Frequently combined with linear processes

Explorative process

- Only goals known, concrete requirements have to be explored
- Stakeholders strongly involved, continuous feedback
- Prioritizing and negotiating requirements to be implemented
- Deadlines and cost constrain functionality
- Typically works only with incremental processes

Prescriptive – Explorative – COTS-driven – 2

COTS-driven process

- System will be implemented with COTS software
- Requirements must reflect functionality of chosen COTS solution
- Requirements need to be prioritized according to importance
- Frequently, only requirements not covered by the COTS solution are specified

COTS (Commercial Off The Shelf) – A system or component that is not developed, but bought as a standard product from an external supplier

Customer-specific vs. Market-oriented

Customer-specific process

- System is ordered by a customer and developed by a supplier for this customer
- Individual persons can be identified for all stakeholder roles
- Stakeholders on customer side are main source for requirements

Market-oriented process

- System is developed as a product for the market
- Prospective users typically not individually identifiable
- Requirements are specified by supplier
- Marketing and system architects are primary stakeholders
- Supplier has to guess/estimate/elicit the needs of the envisaged customers

Typical requirements process configurations

- Participatory: incremental & exploratory & customer-specific
  - Main application case: Supplier and customer closely collaborate; customer stakeholders strongly involved both in specification and development processes

- Contractual: typically linear (sometimes explorative) & prescriptive & customer-specific
  - Main application case: Specification constitutes contractual basis for development of a system by people not involved in the specification and with little stakeholder interaction after the requirements phase

Typical requirements process configuration

- Product-oriented: Incremental & mostly explorative & market-oriented
  - Main application case: An organization specifies and develops software in order to sell/distribute it as a product (or service)

- COTS-aware: [Incremental I linear] & COTS-driven & customer-specific
  - Main application case: The requirements specification is part of a project where the solution is mainly implemented by buying and configuring COTS

Agile requirements process

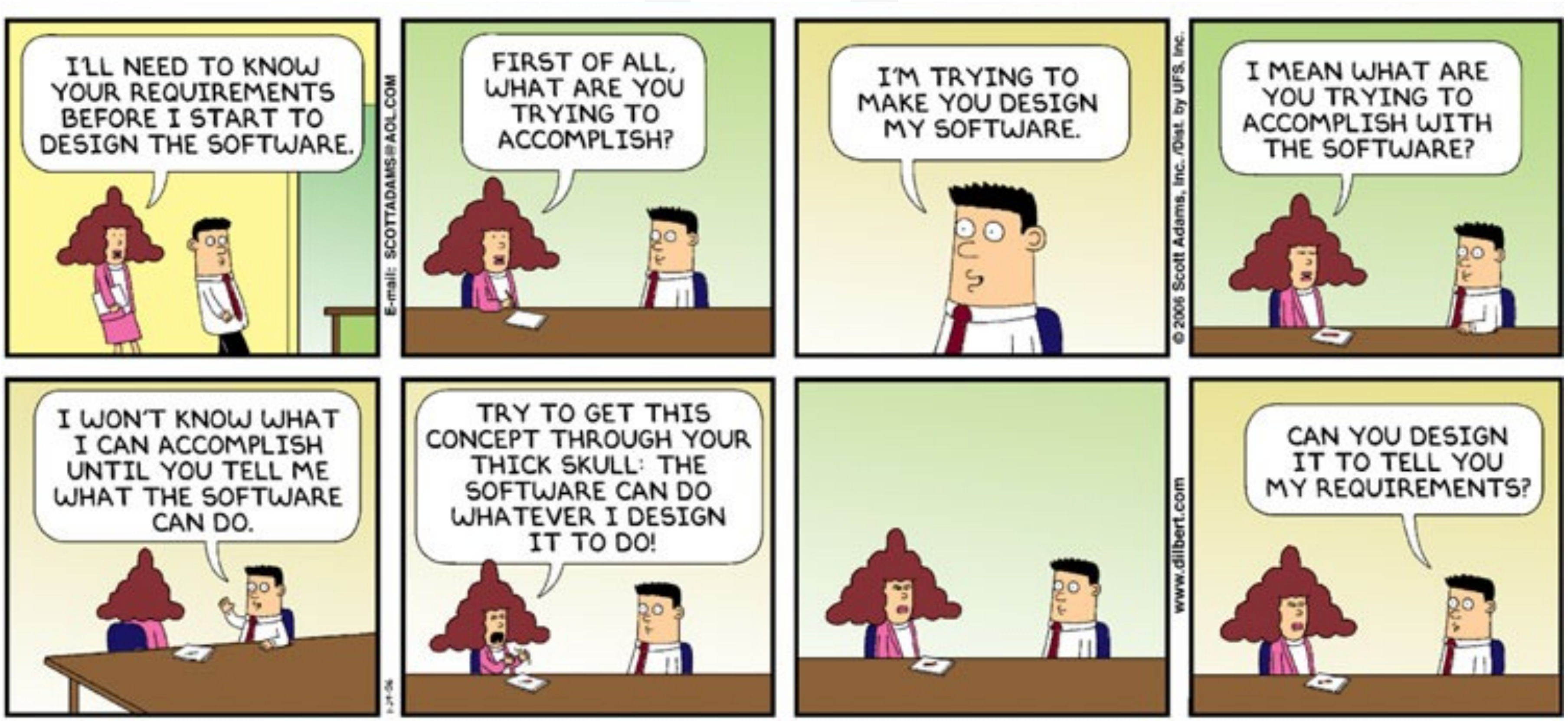
Pushes incrementality and exploration to the extreme

- Fixed-length interations of 1-6 weeks
- Product owner or customer representative always available and has power to make immediate decisions
- Only goals and vision established upfront
- Requirements loosely specified as stories (with details captured in acceptance criteria)
- Use cases or other means used for providing structure & context
- At the beginning of each iteration
  - Customer/product owner prioritizes requirements
  - Developers select what to implement in that iteration
- Short feedback cycle from requirements to deployed system

Characteristics of an “ideal” RE process

- Strongly interactive
- Close and intensive collaboration between
  - Stakeholders (know the domain and the problem)
  - Requirements engineers (know how to specify)
- Very short feedback cycles
- Risk-aware and feasibility-aware
  - Technical risks/feasibility
  - Deadline risks/feasibility
- Careful negotiation / resolution of conflicting requirements
- Focus on establishing shared understanding
- Strives for innovation

7 Requirements elicitation


I'LL NEED TO KNOW YOUR REQUIREMENTS BEFORE I START TO DESIGN THE SOFTWARE.
E-mail: SCOTTADAMS@AOL.COM
FIRST OF ALL, WHAT ARE YOU TRYING TO ACCOMPLISH?
I'M TRYING TO MAKE YOU DESIGN MY SOFTWARE.
© 2006 Scott Adams, Inc. /Dist. by UFS, Inc.
I MEAN WHAT ARE YOU TRYING TO ACCOMPLISH WITH THE SOFTWARE?
I WON'T KNOW WHAT I CAN ACCOMPLISH UNTIL YOU TELL ME WHAT THE SOFTWARE CAN DO.
TRY TO GET THIS CONCEPT THROUGH YOUR THICK SKULL: THE SOFTWARE CAN DO WHATEVER I DESIGN IT TO DO!
www.dilbert.com
CAN YOU DESIGN IT TO TELL YOU MY REQUIREMENTS?

Definition and principles

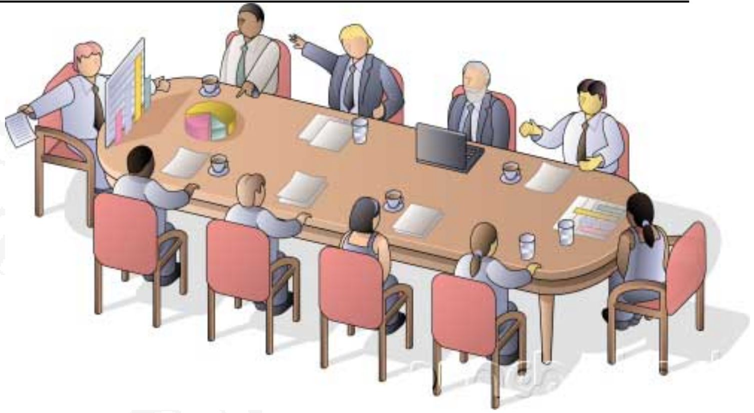
DEFINITION. Requirements elicitation – The process of seeking, capturing and consolidating requirements from available sources. May include the re-construction or creation of requirements.

- ❍ Determine the stakeholders' desires and needs
- ❍ Elicit information from all available sources and consolidate it into well-documented requirements
- ❍ Make stakeholders happy, not just satisfy them
- ❍ Every elicited and documented requirement must be validated and managed
- ❍ Work value-oriented and risk-driven

Information sources

- Stakeholders
- Context
- Observation
- Documents
- Existing systems

Stakeholder analysis



Identify stakeholder roles

End user, customer, operator, project manager, regulator,...

In complex cases: Build model of stakeholder goals, dependencies and rationale

[Yu 1997]
[van Lamsweerde 2001]

Classify stakeholders

- Critical
- Major
- Minor

[Glinz and Wieringa 2007]

Identify/determine concrete persons for each stakeholder role

Context analysis

Photo © Universitätsklinikum Halle (Saale)

Determine the system's context and the context boundary

Identify context constraints

- Physical, legal, cultural, environmental
- Embedding, interfaces

Identify assumptions about the context of your system and make them explicit

Map real world phenomena adequately on the required system properties and capabilities (and vice-versa)

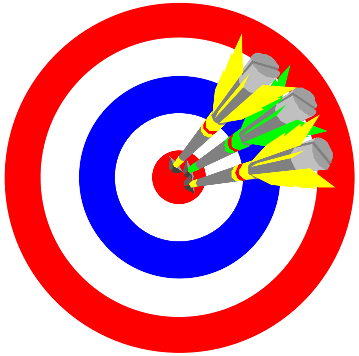
Determine the system scope (cf. Chapter 2.2)

Goal analysis

Knowing your destination is more important than the details of the timetable.

Before eliciting detailed requirements, the general goals and vision for the system to be built must be clear

- ❍ What are the main goals?
- ❍ How do they relate to each other?
- ❍ Are there goal conflicts?



Mini-Exercise

Consider the chairlift access control case study.

(a) Perform a stakeholder analysis.

(b) How can you map the context property that a skier passes an unlocked turnstile to a system property which can be sensed and controlled by the system?

(c) Identify some business goals.

Elicitation techniques

Ask

- ❍ Interview stakeholders
- ❍ Use questionnaires and polls
- ❍ Reply/follow-up to user feedback

Collaborate

- ❍ Hold requirements workshops

Build and play

- ❍ Build, explore and discuss prototypes and mock-ups
- ❍ Perform role playing

[Zowghi and Coulin 2005]
[Dieste, Juristo, Shull 2008]
[Gottesdiener 2002]
[Hickey and Davis 2003]
[Goguen and Linde 1993]

Elicitation techniques – 2

Observe

- Observe stakeholders in their work context

Analyze

- Analyze work artifacts
- Analyze user feedback (problem/bug reports, app reviews, tweets, explicit feedback channels, ...)
- Conduct market studies
- Perform benchmarking

Which technique for what?

Technique	Suitability for			
	Express needs	Demonstrate opportunities	Analyze system as is	Explore market potential
Interviews	+	–	+	o
Questionnaires and polls	o	–	+	+
Workshops	+	o	o	–
Prototypes and mock-ups	o	+	–	o
Role play	+	o	o	–
Stakeholder observation	o	–	+	o
Artifact analysis	o	–	+	–
User feedback analysis	+	–	–	o
Market studies	–	–	o	+
Benchmarking	o	+	–	+

Typical problems

Inconsistencies among stakeholders in

- needs and expectations
- terminology

Stakeholders who know their needs, but can’t express them

Stakeholders who don’t know their needs

Stakeholders with a hidden agenda

Stakeholders thinking in solutions instead of problems

Stakeholders frequently neglect attributes and constraints

➥ Elicit them explicitly

Who should elicit requirements?

- Stakeholders must be involved
- Domain knowledge is essential
  - Stakeholders need to have it (of course)
  - Requirements engineers need to know the main domain concepts
  - A “smart ignoramus” can be helpful [Berry 2002, Sect. 7]
- Don’t let stakeholders specify themselves without professional support
- Best results are achieved when stakeholders and requirements engineers collaborate

Eliciting functional requirements

- Who wants to achieve what with the system?
- For every identified function
  - What’s the desired result and who needs it?
  - Which transformations and which inputs are needed?
  - In which state(s) shall this function be available?
  - Is this function dependent on other functions?
- For every identified behavior
  - In which state(s) shall the system have this behavior?
  - Which event(s) lead(s) to this behavior?
  - Which event(s) terminate(s) this behavior?
  - Which functions are involved?

Eliciting functional requirements – 2

- For every identified data item
  - What are the required structure and the properties of this item?
  - Is it static data or a data flow?
  - If it’s static, must the system keep it persistently?
- Analyze mappings
  - How do real world functions/behavior/data map to system functions/behavior/data and vice-versa?
- Specify normal and exceptional cases

Eliciting quality requirements

Stakeholders frequently state quality requirements in qualitative form:

"The system shall be fast."

"We need a secure system."

Problem: Such requirements are

- Ambiguous
- Difficult to achieve and verify

❍ Classic approach:

- Quantification ➔ ⊕ measurable ⊖ maybe too expensive
- Operationalization ➔ ⊕ testable ⊖ implies premature design decisions

New approach to eliciting quality requirements

[Glinz 2008]

Represent quality requirements such that they deliver optimum value

Value of a requirement = benefit of development risk reduction
minus cost for its specification

- Assess the criticality of a quality requirement
- Represent it accordingly
- Broad range of possible representations

The range of adequate representations

Situation	Representation	Verification
1. Implicit shared understanding	Omission	Implicit
2. Need to state general direction Customer trusts supplier	Qualitative	Inspection
3. Sufficient shared understanding to generalize from examples	By example	Inspection, (Measurement)
4. High risk of not meeting stake-holders' desires and needs	Quantitative in full	Measurement
5. Somewhere between 2 and 4	Qualitative with partial quantification	Inspection, partial measurement

Eliciting performance requirements

Things to elicit

- ❍ Time for performing a task or producing a reaction
- ❍ Volume of data
- ❍ Throughput (data transmission rates, transaction rates)
- ❍ Frequency of usage of a function
- ❍ Resource consumption (CPU, storage, bandwidth, battery)
- ❍ Accuracy (of computation)

Eliciting performance requirements – 2

- What's the meaning of a performance value:
  - Minimum?
  - Maximum?
  - On average?
  - Within a given interval?
  - According to some probability distribution?
- How much deviation can be tolerated?

Eliciting specific quality requirements

- Ask stakeholders explicitly
- A quality model such as ISO/IEC 25010:2011(formerly ISO/IEC 9126) can be used as a checklist
- Quality models also help when a specific quality requirement needs to be quantified

Eliciting constraints

- Ask about restrictions of the potential solution space
  - Technical, e.g., given interfaces to neighboring systems
  - Legal, e.g., restrictions imposed by law, standards or regulations
  - Organizational, e.g. organizational structures or processes that must not be changed by the system
  - Cultural, environmental, ...
- Check if a requirement is concealed behind a constraint
  - Constraint stated by a stakeholder: “When in exploration mode, the print button must be grey.”
  - Actual requirement: “When the system is used without a valid license, the system shall disable printing.”

Mini-Exercise

Consider the chairlift access control case study.

(a) Which technique(s) would you select to elicit requirements from the chairlift ticket office clerks?

(b) How, for example, can you achieve consensus among the ski resort management, the technical director of chairlifts, the ticket office clerks, and the service employees?

(c) Identify some constraints for the chairlift access control system.

Analysis of elicited information

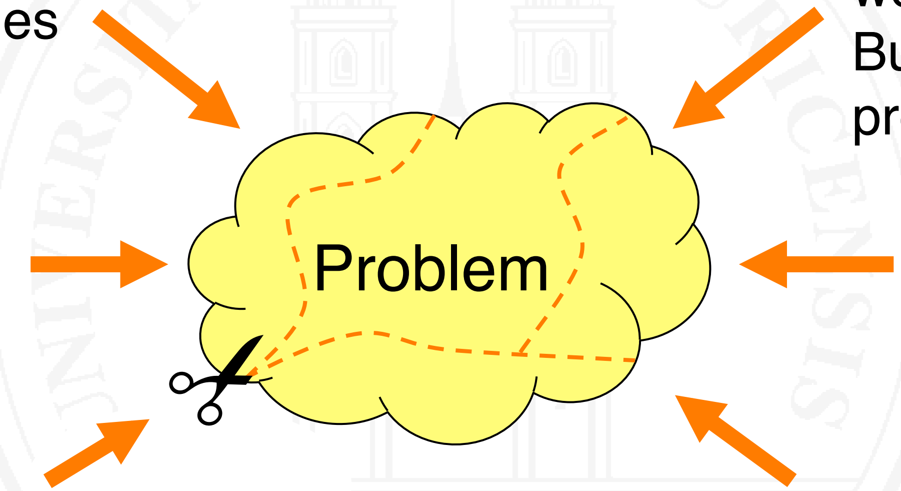
Structure-oriented

Process-oriented

Analyze terminology / domain properties
Build glossary

Analyze processes / workflows
Build activity / process models

Analyze business and data objects
Build object and class models



Analyze dynamic system behavior
Build behavior model

Decompose problem
Build hierarchical structure

Analyze actor-system interaction
Build scenarios / use cases

Note: requirements are about a future state of affairs; analyze the current state only when necessary

Documenting elicited requirements

Build specification incrementally and continuously

Document requirements in small units

End over means: Result → Function → Input

Consider the unexpected: specify non-normal cases

Quantify critical attributes

Document critical assumptions explicitly

Avoid redundancy

Build a glossary and stick to terminology defined in the glossary