L4 – Interpretation (continued) and L6 – Negotiation

DAT232/DIT285 Advanced Requirements Engineering

Eric Knauss
eric.knauss@cse.gu.se

September 23, 2025

Visit **gosocrative.com** and enter room name REQENG

Lecture starts at 15:15

You are welcome to share a short break with us and discuss/ask questions

Outline

Housekeeping

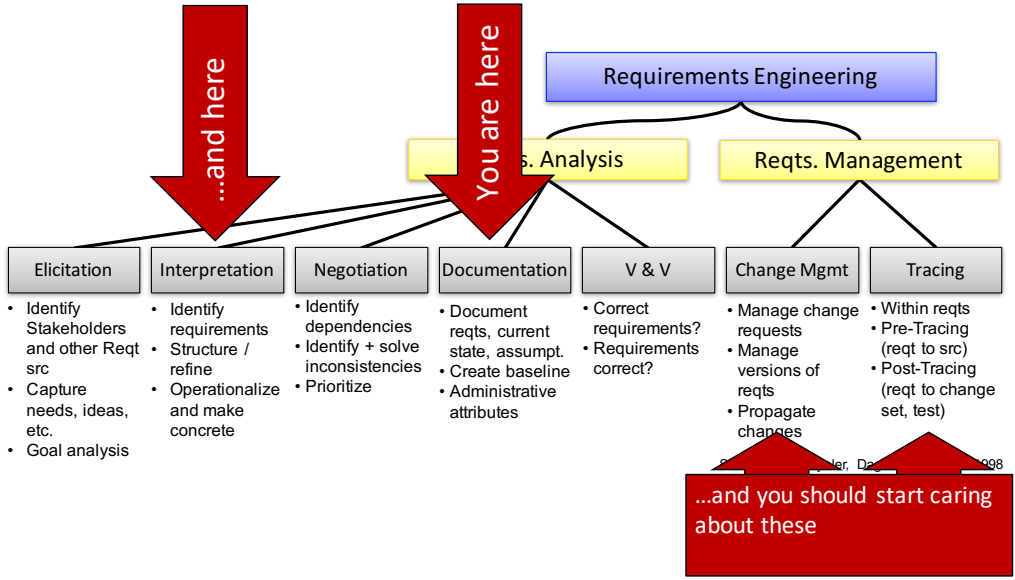
- R1 submitted. We will give you feedback as soon as possible (during the week)
- This week: Documentation workshop
- Self-assessment: Assignment (and Form) is now open and will be discussed during ICC workshop next week
- Re-work if single workshop is missed: Will be assigned shortly
- Idea behind 3 releases: Do, assess, feedback, reflect, do better... we will only grade at R3. R1 and R2 are passed when submitted (no reason to do rework).

Feedback from Socrative

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

Hide Answers | Show Names | 9/51 Students Answered

socrative
going beyond the slides added value to the lecture which was nice. it was informative and easy to follow along. No complaints
Clear examples are so useful!!
Interactiveness and explicit examples were great
-
In L3 we had a slide, "Things to cover from L2", showing "Must know" and "Should know". I would like to have such a slide in every lecture.
I liked your explainations of the questions and I wish to learn about UML and similar.
One thing i like about this course the questions that are there. Allow for active learning.
Very descriptive material and on point explaination by professor knauss.


...and here
You are here
Requirements Engineering
s. Analysis
Reqts. Management
Elicitation
Interpretation
Negotiation
Documentation
V & V
Change Mgmt
Tracing
• Identify Stakeholders and other Reqt src
• Capture needs, ideas, etc.
• Goal analysis
• Identify requirements
• Structure / refine
• Operationalize and make concrete
• Identify dependencies
• Identify + solve inconsistencies
• Prioritize
• Document reqts, current state, assumpt.
• Create baseline
• Administrative attributes
• Correct requirements?
• Requirements correct?
• Manage change requests
• Manage versions of reqts
• Propagate changes
• Within reqts
• Pre-Tracing (reqt to src)
• Post-Tracing (reqt to change set, test)
...and you should start caring about these

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, ...)

Interpretation for ...Interpretation:

K1,K2,K3	Explain common challenges of interpretation and documentation, describe practices of interpretation and documentation of reqts
K4	Be aware of current research challenges in interpretation
S1-S3	Apply this knowledge to your project
J1-J3	Look out for opportunities to get beyond course literature
J4	Reflect on how your background changes your view on documentation (different documentation for games, data-intense projects, with technology/management focus?)

Outline

What is requirements interpretation

Activity	*Explanation*
Identify requirements	Elicitation provides us with candidate requirements. Those things mentioned by our stakeholders: Are those really needed to solve the problem?
Structure and refine	Go through the candidate requirements and analyse them.
Operationalize and make concrete	It is easy to agree with vague and abstract requirements. The system should be usable. Sure. But how usable and at what cost?

Quality criteria

What is requirements interpretation

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

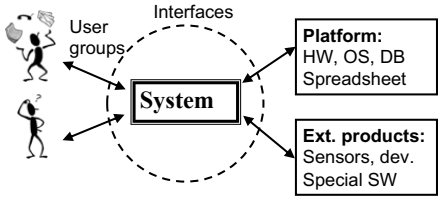
Correct	Each requirement reflects a need.
Complete	All necessary requirements included.
Unambiguous	All parties agree on meaning.
Consistent	All parts match, e.g. E/R and event list.
Ranked for importance and stability	Priority and expected changes per requirement.
Modifiable	Easy to change, maintaining 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.

How to fulfil those quality criteria?

Common approach: **Structured, top-down**

For everything mentioned by stakeholders:

1. Is it a requirement? No: Add as information, if helpful. Yes: Continue
2. What type of requirement? Functional? Data? Quality? Other? Write at correct place.
3. Check, whether each part has been filled out and whether information is consistent.
4. Add tracelinks to indicate dependencies.



Data requirements:

System state: Database, comm. states
Input/output formats

Functional requirements, each interface:

Record, compute, transform, transmit
Theory: F(input, state) -> (output, state)
Function list, pseudocode, activity diagram
Screen prototype, support tasks xx to yy

From: Soren Lauesen: Software Requirements

Quality reqs:

Performance
Usability
Maintainability
. . .

Other deliverables:

Documentation
Install, convert, train . . .

Managerial reqs:

Delivery time
Legal
Development process . . .

Helping the reader:

Business goals
Definitions
Diagrams . . .

Functional requirements (FR)

- What the system shall do
- Often intended to be implemented as a whole or else not implemented at all
- Often regards input/output data and functions that process the input data to produce the output

Non-functional requirements / Quality requirements (QR)

- How good the system shall do it
- Often measured on a scale
- Often put constraints on the system (or the development process)
- Often cross-cutting; may impact many functions

- Performance
- Reliability
- Usability
- Safety, Security
- Interoperability
- Maintainability
- Functionality (!)
- ...

Functional requirements (FR)

- What the system shall do
- Often intended to be implemented as a whole or else not implemented at all
- Often regards input/output data and functions that process the input data to produce the output

Non-functional requirements / Quality requirements (QR)

- How good the system shall do it
- Often measured on a scale
- Often put constraints on the system (or the development process)
- Often cross-cutting; may impact many functions
- Performance
- Reliability
- Usability
- Functionality (!)
- ...

However ...

...the distinction is not always so black and white

FR & QR are often tightly coupled

In practice it is often difficult to separate functional and quality requirements as quality requirements often are manifested into extra functionality.

Example

Quality requirement on security requires a log-in function.

→ `socrative.com`, Room REQENG, Q1

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 requirementis 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.

!?!

Discuss!

Which quality requirements of a word processor do you appreciate?

→ `socrative.com`, Room REQENG, Q2

Difficult trade-offs among QR

Quality requirements can counteract each other.
Common examples:

- Higher performance → lower maintainability
- Higher security → lower usability

Requires carefully considered trade-offs!

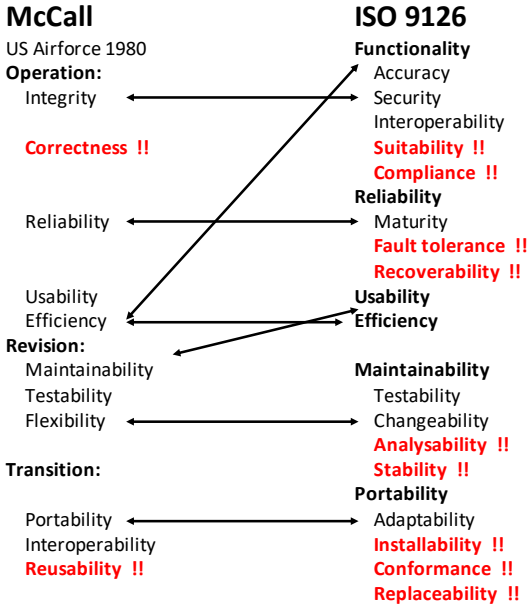
Quality requirements often input for architectural decisions.

Why is it difficult to manage NFR?

- Trade-offs
- Often not binary, but can be fulfilled to degrees
- Negotiation?!
- Cross-cutting (since relating to architecture)
  - May impact many functional requirements!
- How to test them?
- Unclear / many taxonomies

Fig 6.1 Quality factors

From: Soren Lauesen: Software Requirements



Use as check lists

UNIVERSITY OF GOTHENBURG

CHALMERS UNIVERSITY OF TECHNOLOGY

Current Taxonomies

ISO standards improving [ISO/IEC25010, 2022, ISO/IEC25019, 2022]

Problems with Taxonomies

- Distinguish product quality and quality in use (but poorly defined as a *either-or*)
- Safety, Privacy, Sustainability, Explainability missing

Glinz et al. suggest to start from the list to the right [Glinz et al., 2023]

Towards a modern Quality Framework

[Glinz et al., 2023]

Characteristic	**Subcharacteristics**
Functionality	Structure and data, Function and flow, State and behavior, Context and boundary
Performance	Time, Volume, Frequency, Throughput, Resource consumption
Usability	Learnability, Ease of use, User assistance, Explainability
Sustainability	Environmental, Social Reliability
Security	Availability, Fault tolerance, Recoverability Confidentiality, Integrity, Non-repudiation, Accountability, Authenticity, Resistance
Safety	Preventability, Resilience
Privacy	Data sovereignty, Data minimization, Anonymity, Non-disclosure
Maintainability	Modifiability, Reusability
Portability	Adaptability, Installability, Scalability
Compatibility	Co-existence, Interoperability

Towards a modern Quality Framework

[Glinz et al., 2023]

Glinz et al. also propose the following emergent properties [Glinz et al., 2023]

On Quality In Use

- Pragmatic view: It has an influence on operationalization
- Can it be tested in a static way? At runtime? In use?

Characteristic	Subcharacteristics
Compliance	Functionality, Performance, Sustainability, Reliability, Security, Safety, Privacy
Dependability	Reliability, Security, Safety, Privacy
Efficiency	Functionality, Performance, Usability
User experience	Functionality, Performance, Usability, Dependability

Principle ideas

Non-functional requirements

Make quality measurable (e,g, Quality Models or Planguage)

Goal 1: Progress control

- Beyond status of realized functions
- Quality characteristics
  - Complexity
  - Maintainability
  - Robustness

Goal 2: Quality improvement (of method, tool, product, etc.)

- Predict improvement by specific activities
- Check if improvement goals were achieved

Quality models

Characteristics
(Functionality, Usability, ...)

Sub-Characteristics
(Suitability, Learnability, ...)

Generic / abstract

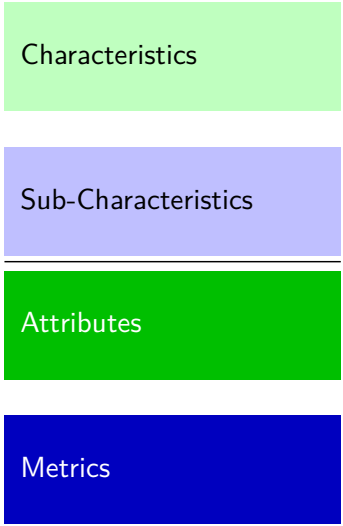
Attributes

Organization specific

Metrics
(Internal, External, Quality in Use)

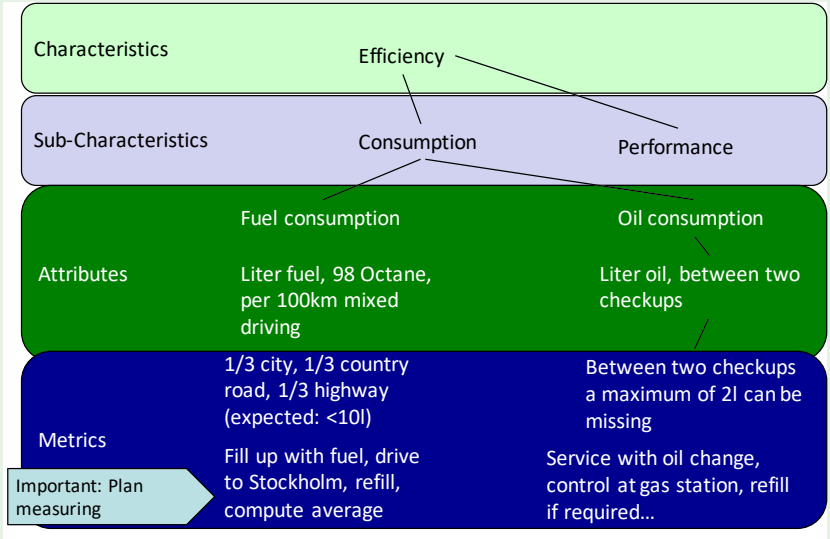
Measurable

Quality models



- Re-use existing schema (see previous slides or ISO standard)
- Individual refinement and prioritization
  - iterative, feedback from domain- and technical experts
- Which aspects should be measured?
  - Which aspects can be accessed?
  - Product or process?
- How will be measured
  - Established metrics
  - New, specific indicators
  - Expected results

Example


Characteristics
Efficiency
Sub-Characteristics
Consumption
Performance
Attributes
Fuel consumption
Liter fuel, 98 Octane, per 100km mixed driving
Oil consumption
Liter oil, between two checkups
Metrics
1/3 city, 1/3 country road, 1/3 highway (expected: <10l)
Fill up with fuel, drive to Stockholm, refill, compute average
Between two checkups a maximum of 2l can be missing
Service with oil change, control at gas station, refill if required…
Important: Plan measuring

Other Principles

- Focus on few that really matter
- Clearly indicate assumptions / ability to negotiate
- Provide clear tracing between NFR and FR

Fig 6.2 Quality grid

Quality factors for Hotel system	Critical	Impor-tant	As usual	Unim-portant	Ignore
Operation					
Integrity/security			X		
Correctness			X		
Reliability/availab.		1			
Usability		2			
Efficiency			X		
Revision					
Maintainability			X		
Testability			X		
Flexibility			X		
Transition					
Portability					X
Interoperability	3			4	
Reusability					X
Installability		5			

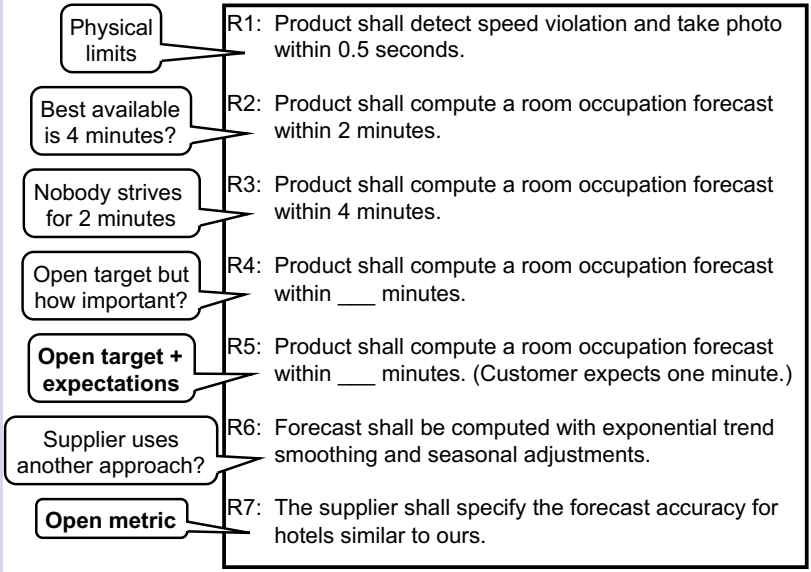
Concerns:

1. Hard to run the hotel if system is down. Checking in guests is impossible since room status is not visible.
2. We aim at small hotels too. They have less qualified staff.
3. Customers have many kinds of account systems. They prioritize smooth integration with what they have.
4. Integration with spreadsheet etc. unimportant. Built-in statistics suffice.
5. Must be much easier than present system. Staff in small hotels should ideally do it themselves.

From: Soren Lauesen: Software Requirements

→ socrative.com, Room REQENG, Q3

Fig 6.3A Open metric and open target



→ socrative.com, Room REQENG, Q4

Fig 6.3B Planguage version of target etc.

Forecast speed [Tag]: How quickly the system completes a forecast report [Gist]

Scale: average number of seconds from pushing button, to report appearing.

Meter: Measured 10 times by a stopwatch during busy hours in hotel reception.

Must: 8 minutes, because the competitive system does it this fast.

Plan: ____ (supplier, please specify).

Wish: 2 minutes.

Past: Done as batch job taking about an hour.

From: Soren Lauesen: Software Requirements

Fig 6.4 Capacity and accuracy requirements

Capacity requirements:

R1: The product shall use < 16 MB of memory even if more is available.

R2: Number of simultaneous users < 2000

R3: Database volume:
#guests < 10,000 growing 20% per year
#rooms < 1,000

R4: Guest screen shall be able to show at least 200 rooms booked/occupied per day, e.g. for a company event with a single “customer”.

Accuracy requirements:

R5: The name field shall have 150 chars.

R6: Bookings shall be possible at least two years ahead.

R7: Sensor data shall be stored with 14 bit accuracy, expanding to 18 bits in two years.

R8: The product shall correctly recognize spoken letters and digits with factory background noise ___ % of the time. Tape B contains a sample recorded in the factory.

Fig 6.5A Performance requirements

Performance requirements:

R1: Product shall be able to process 100 payment transactions per second in peak load.

R2: Product shall be able to process one alarm in 1 second, 1000 alarms in 5 seconds.

R3: In standard work load, CPU usage shall be less than 50% leaving 50% for background jobs.

R4: Scrolling one page up or down in a 200 page document shall take at most 1s. Searching for a specific keyword shall take at most 5s.

R5: When moving to the next field, typing must be possible within 0.2s. When switching to the next screen, typing must be possible within 1.3s. Showing simple report screens, less than 20s. (Valid for 95% of the cases in standard load)

R6: A simple report shall take less than 20s for 95% of the cases. None shall take above 80s. (UNREALISTIC)

Cover all product functions?

Fig 6.6A Usability

Usability requirements?

R1: System shall be easy to use??

R2: 4 out of 5 new users can book a guest in 5 minutes, check in in 10 minutes, . . . *New user* means . . . Training . . .

Achieving usability

- Prototypes (mockups) before programming.
- Usability test the prototype.
- Redesign or revise the prototype.

Easier programming. High customer satisfaction.

Defect types

Program error: Not as intended by the programmer.

Missing functionality: Unsupported task or variant.

Usability problem: User cannot figure out . . .

Fig 6.6B Usability problems

Examples of usability problems

P1: User takes long time to start search. Doesn't notice "Use F10". Tries many other ways first.

P2: Believes task completed and result saved. Should have used *Update* before closing.

P3: Cannot figure out which discount code to give customer. Knows which field to use.

P4: Crazy to go through 6 screens to fill 10 fields.

Problem classification

Task failure: Task not completed - or believes it is completed.

Critical problem: Task failure or complaints that it is cumbersome.

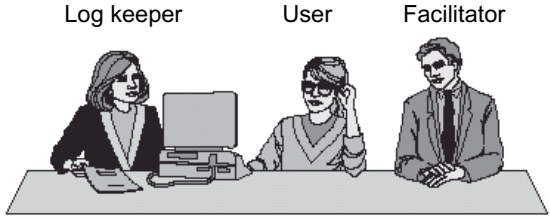
Medium problem: Finds out solution after lengthy attempts.

Minor problem: Finds out solution after short attempts

Fig 6.6C Usability test & heuristic evaluation

Usability test
Realistic introduction
Realistic tasks

Note problems
- Observe only or
- Think aloud & ask



Heuristic evaluation
Expert's predicted problems
≅ Inspection/Review

Usability test:
Cover all tasks?
Mockups find same problems as test with final system?

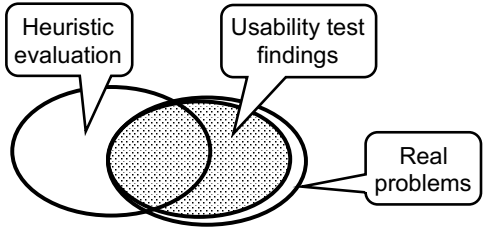


Fig 6.7(A) Usability requirements

	Risk Cust.	Risk Suppl
Problem counts R1: At most 1 of 5 novices shall encounter critical problems during tasks Q and R. At most 5 medium problems on list.		
Task time R2: Novice users shall perform tasks Q and R in 15 minutes. Experienced users tasks Q, R, S in 2 minutes.		
Keystroke counts R3: Recording breakfast shall be possible with 5 keystrokes per guest. No mouse.		
Opinion poll R4: 80% of users shall find system easy to learn. 60% shall recommend system to others.		
Score for understanding R5: Show 5 users 10 common error mesages, e.g. *Amount too large*. Ask for the cause. 80% of the answers shall be correct.		

→ socrative.com, Room REQENG, Q5

Fig 6.7(A) Usability requirements

	Risk Cust.	Risk Suppl
Problem counts R1: At most 1 of 5 novices shall encounter critical problems during tasks Q and R. At most 5 medium problems on list.		
Task time R2: Novice users shall perform tasks Q and R in 15 minutes. Experienced users tasks Q, R, S in 2 minutes.		
Keystroke counts R3: Recording breakfast shall be possible with 5 keystrokes per guest. No mouse.		
Opinion poll R4: 80% of users shall find system easy to learn. 60% shall recommend system to others.		
Score for understanding R5: Show 5 users 10 common error mesages, e.g. *Amount too large.* Ask for the cause. 80% of the answers shall be correct.		

State of the art

Reasons for difficulties in prioritizing QR [Svensson et al., 2011]:

- Elicitation of QR
- Lack of well specified QR
- Quantify QR
- What is good enough?
- Knowledge about QR

Quality Requirements challenge in market-driven RE

Systematic prioritization of FEATURES is state-of-art in roadmapping and platform/product scoping

...but...

Prioritisation of QUALITIES is often handled ad hoc with no specific support for roadmapping
One FR imply many different qualities. How to scope both FR and QR together? See [Svensson et al., 2012].

Other hot research topics

- Privacy [Deng et al., 2011] and transparency [Hosseini et al., 2016]
- Ethics, e.g. with respect to AI
- Our own works
  - Identify security-relevant requirements [Houmb et al., 2010, Schneider et al., 2012]
  - Manage safety requirements in agile system development [Kasauli et al., 2018]
  - Manage non-functional requirements in large-scale agile [Knauss et al., 2017]

Outline

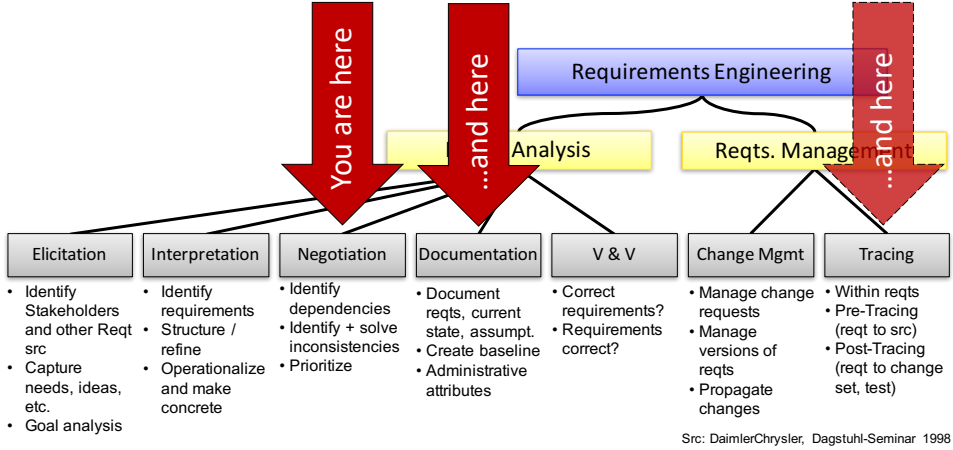
L4 – Interpretation (continued) and L6 – Negotiation

DAT232/DIT285 Advanced Requirements Engineering

Eric Knauss
eric.knauss@cse.gu.se

September 23, 2025

Requirements Engineering



Src: DaimlerChrysler, Dagstuhl-Seminar 1998

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, ...)

Interpretation for ...Negotiation:

K1,K2,K3	Explain common challenges of negotiation, describe practices of negotiation of reqts
K4	Be aware of current research challenges in negotiation
S1-S3	Apply this knowledge to your project
J1-J3	Look out for opportunities to get beyond course literature
J4	Reflect on how your background changes your view on negotiation (different documentation for games, data-intense projects, with technology/management focus?)

Take-aways for your project

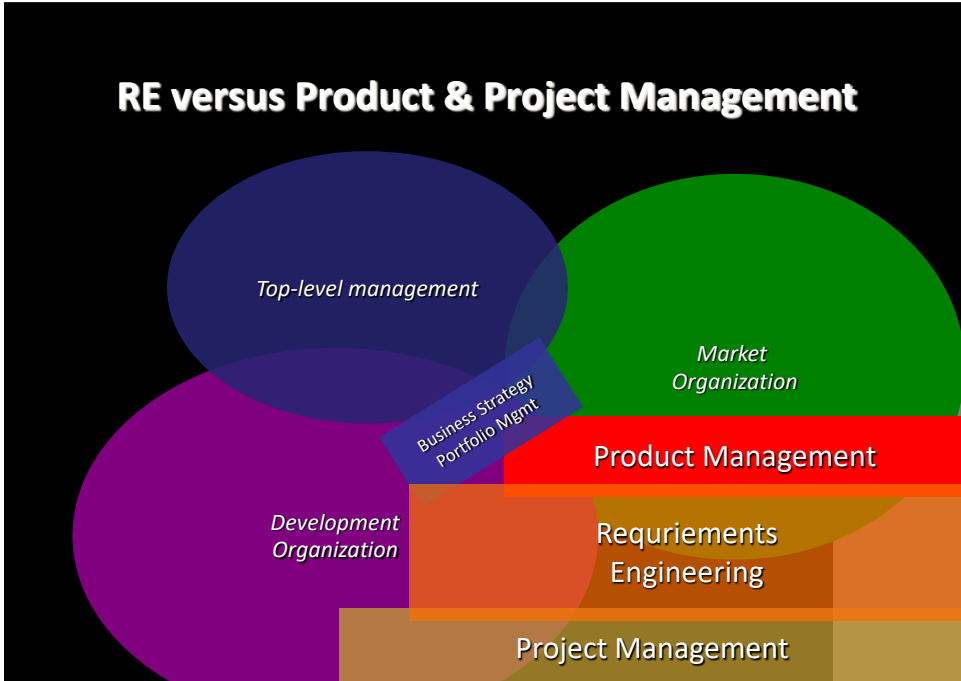
Identify your prioritization objects

- Features
- Domain events
- Main tasks
- Quality attributes

Explore different methods for prioritization!

Mindful about release planning

- Plan the first release to be minimal
- Exploit inter-dependencies
- Do not apply methods from [RP], since you are lacking information about cost/resources.

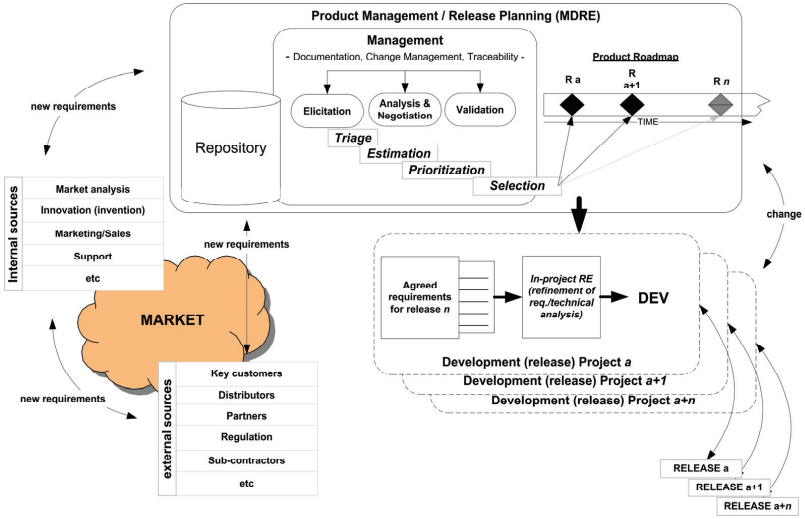

RE versus Product & Project Management
Top-level management
Market Organization
Business Strategy Portfolio Mgmt
Product Management
Development Organization
Requriements Engineering
Project Management

Book Chapter [MDRE]

Market-Driven Requirements Engineering for Software Products

B. Regnell, S. Brinkkemper

Engineering and Managing Software Requirements, Eds. A. Aurum and C. Wohlin, Springer, ISBN 3-540-25043-3, 2005


Product Management / Release Planning (MDRE)
Management
- Documentation, Change Management, Traceability -
Elicitation
Analysis & Negotiation
Validation
Repository
Triage
Estimation
Prioritization
Selection
Product Roadmap
R a
R a+1
R n
TIME
new requirements
Internal sources
Market analysis
Innovation (invention)
Marketing/Sales
Support
etc
new requirements
MARKET
new requirements
external sources
Key customers
Distributors
Partners
Regulation
Sub-contractors
etc
change
Agreed requirements for release n
In-project RE (refinement of req./technical analysis)
DEV
Development (release) Project a
Development (release) Project a+1
Development (release) Project a+n
RELEASE a
RELEASE a+1
RELEASE a+n

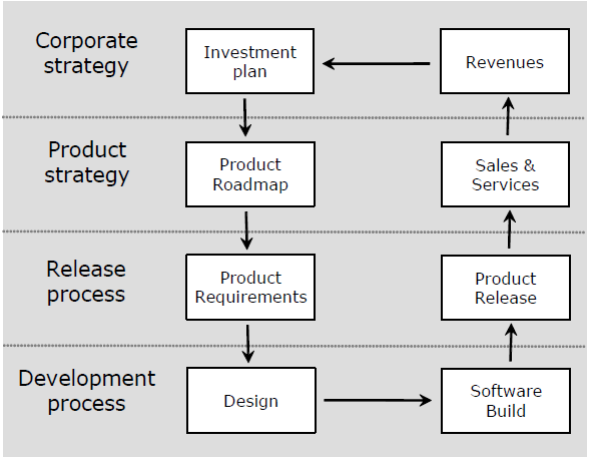
Different types of products

Table 13.1. Examples of variants of hardware and software products.

	Pure Hardware	*Embedded Systems (HW+SW)*	*Pure Software*
Generic	Note sticks	Mobile phone	Firewall
Customized	Office furniture	Customized car	Enterprise resource planning systems
Customer-Specific	Portrait painting	Military vehicle	Web Site

- The distinction is not strict: the same org. often combines several types, e.g. generic+customized
- Sometimes products evolve from customer specific to generic.

The investment cycle



!?!

Discuss!

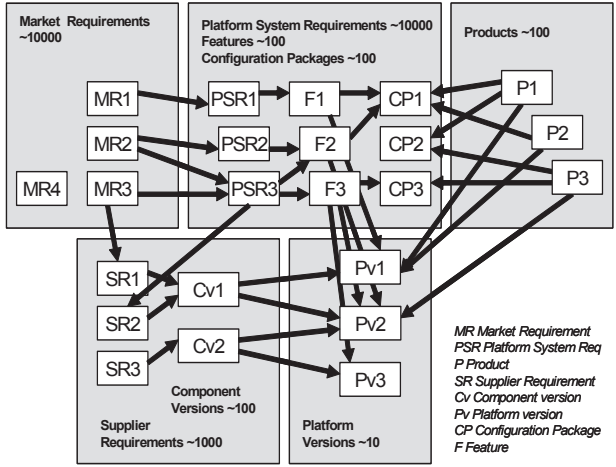
What are the main differences between market-driven RE (MDRE) and customer-specific RE (bespoke RE)?

→ `socrative.com`, Room REQENG, Q6

Specific characteristics of Market-Driven RE

- Basic goals
  - Profit, margins, future revenue
  - Market shares
- Inventing reqs (not just collect)
- Stakeholders
  - Market segments with potential customers
  - Competitors and confidentiality
- Planning
  - Time to market
  - Multiple releases
- Continuous inflow of reqs

[MDRE]


Market Requirements ~10000
Platform System Requirements ~10000
Features ~100
Configuration Packages ~100
Products ~100
MR1
MR2
MR3
MR4
PSR1
PSR2
PSR3
F1
F2
F3
CP1
CP2
CP3
P1
P2
P3
SR1
SR2
SR3
Cv1
Cv2
Component Versions ~100
Supplier Requirements ~1000
Pv1
Pv2
Pv3
Platform Versions ~10
MR Market Requirement
PSR Platform System Req
P Product
SR Supplier Requirement
Cv Component version
Pv Platform version
CP Configuration Package
F Feature

Some hard challenges in MDRE

1. Balancing market pull and technology push
2. Chasm between marketing and development
3. Organizational instability and market turbulence
4. Simple tools for basic needs
5. Requirements dependencies
6. Cost-value-estimation and release planning
7. Overloaded Requirements Management

Decisions outcomes in MDRE

		Decision	
		Selected	*Rejected*
Requirements Quality	*alfa*	*A* Correct selection ratio	*B* Incorrect selection ratio
	beta	*C* Incorrect selection ratio	*D* Correct selection ratio

Product Quality: $Q_p = A/(A+C)$

Decision Quality: $Q_d = (A+D)/(A+B+C+D)$

Finding the golden grains despite uncertain estimates of value and cost

Figure 13.1 (a) Cost-Value Diagram with alfa-requirements (filled) and beta-requirements (empty).

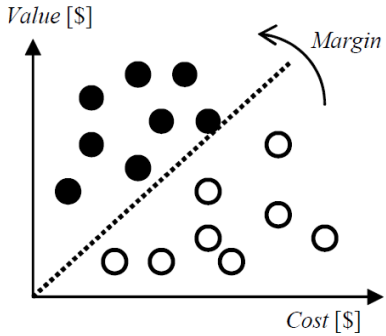
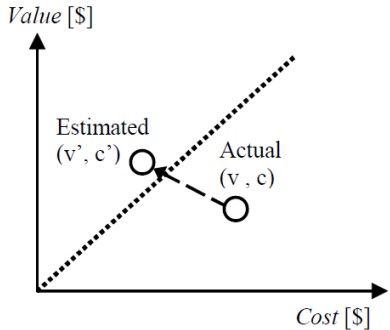


Figure 13.1 (b) Estimated values are differing from actual values causing wrong selection decision.



Requirements Prioritisation [PRIO]

Discussion Prioritization

- Consider a previous project that you participated in:
  - Why was prioritization needed?
  - When did you prioritize?
  - How did you prioritize?
  - Who (which roles) did the prioritization?
  - What types of requirements (level etc) did you prioritize?

Why prioritize?

- To focus on the most important issues
- To find high & low priority requirements
- To implement requirements in a good order
- To save time and money

Prioritization challenges

- Finding a good abstraction level
- Combinatorial explosion
- Inter-dependencies
- Not easy to predict the future
- Power and politics

What do you need to do?

- Select prioritization criteria
- Select prioritization objects
- Structure and group:
  appropriate level and reasonable number
- Do the actual prioritization:
  decide priorities for each criteria and object
- Visualize, discuss, iterate, …

Objects & Criteria

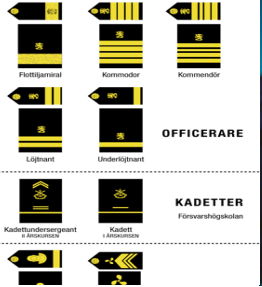
- Typically features at high level that can be selected or de-selected
- Example criteria.
  - Customer value
  - Development cost
  - Development lead-time
  - Engineering Risk
  - Fitness to market window
  - Fitness to brand
  - Volatility
  - …
- Maximize / Minimize

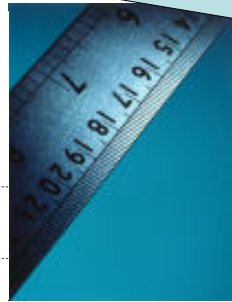
When to prioritize?

- At decision points (toll gates), e.g.
  - Project start
  - Start of construction
  - Release planning
  - Increment planning
- When big changes occur
- Iteratively with appropriate intervals

Prioritization scales

Which is best? Trade-off between data quality and expressive power of scale!





Categorization

e.g.: must, ambiguous, volatile

Partition in groups without greater-less relations

Ordinal scale

e.g.: more expensive, higher risk, higher value

Ranked list
A>B

Ratio scale

ex: $, h,
% (relative)

Numeric leations:
A=2*B

Typical industry praxis

- Numerical assignment ,e.g. 1-5
- Problem:
  - Reqs are not confronted to each other
  - What should go out when new req wants in?
  - Everything tends to be important; hesitation to decide

Different techniques for requirements prioritization

- Priority grouping, grades
  - Ordinal scale; quick & easy; risk that all reqs are important as they are not challenged against each other; may be misinterpreted as ratio scale (Even if ”4” not necessarily is twice as much as ”2”)
- 100$-test
  - Ratio scale, quick and easy, risk of shrewd tactics
- Ranking (sorting)
  - Ordinal scale, pairwise comparison, easy and rather quick
- Top-ten
  - Ordinal scale, very quick and simple, gives a rough estimate on a limited set of reqs
- Analytical Hierarchy Process (AHP)
  - Ordinal scale, pairwise comparison, tool is needed, redundancy gives measure of consistency

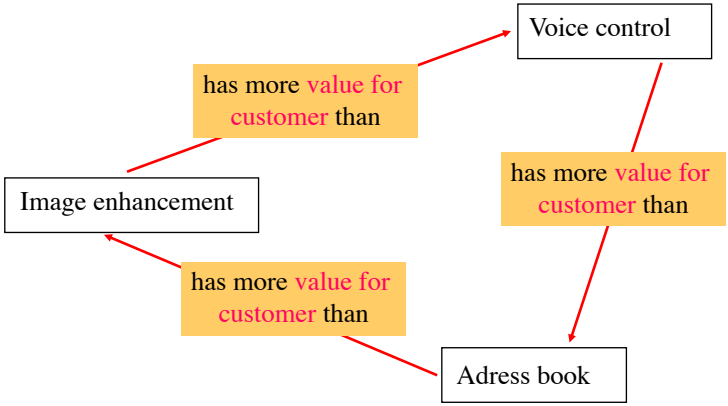
100$-test

- Each stakeholder is given a fictitious amount of “money” to distribute over reqs to reflect the assessment of a criteria from that stakeholder’s point of view:
- Strengths
  - Easy to understand
  - Ratio scale
- Weaknesses
  - Opens up for shrewd tactics
  - No internal consistency analysis
  - The requirements are systematically compared in pairs

Analytical Hierarchy Process (AHP)

- **Developed by Thomas L. Saaty in the 80'ies**
- **Commonly known technique in decision theory**
- **Pairwise comparison of all pairs on a 9-level ratio scale**
- **Matrix calculations gives**
  - **priorities and**
  - **a measure of the degree of consistency among comparisons**
- **Summary of the steps in AHP**
  1. Do all n(n-1)/2 pairwise comparisons
  2. Create priority matrix:
     1. Ones in the diagonal
     2. Mirror inverse: $n_{ij} = 1/n_{ji}$
  3. Divide each element by the sum of its columns
  4. Create a vector with row sums
  5. Divide with the number of reqs -> priority vector
  6. Multiply the priority matrix with the priority vector
  7. Divide each element of the result with the priority vector
  8. L= average of all elements
     CI=(L-n)/(n-1) "consistency index"
     CR=CI/RI "consistency ratio" (where RI depends on n)

Tools can help you find inconsistent comparisons



Outline

Take-aways for your project

Identify your prioritization objects

- Features
- Domain events
- Main tasks
- Quality attributes

Explore different methods for prioritization!

Mindful about release planning

- Plan the first release to be minimal
- Exploit inter-dependencies
- Do not apply methods from [RP], since you are lacking information about cost/resources[1].

[1] Plus, we have removed advanced release planning and [RP] from this slide deck

Criteria 4/5 (Prioritization)

Project assessment

ID	*Where checked*	*Criteria*	*F*	*3*	*4*	*5*
p.1	Requirements Document / Experience Report (prioritization)	More than one suitable prioritization technique applied.				
p.2	Requirements Document / Experience Report (prioritization)	Both functional and non functional requirements are prioritized in a consistent way.				
p.3	Requirements Document / Experience Report (prioritization)	Requirements prioritization is consistent across requirements abstraction levels.				
p.4	Requirements Document / Experience Report (prioritization, reflection)	Integrates and reflects on several prioritization techniques that are applied iteratively to different types of requirements.				
p.5	Requirements Document / Experience Report (prioritization, reflection)	Stakeholder priorities are weighted and combined with clear rationale.				
p.6	Requirements Document / Experience Report (prioritization, reflection)	Priorities guide improvements in specification quality for a targeted subset of requirements, based on reflective experiences.				
p.7	Experience Report (reflection)	Reflects on prioritization experiences.				
p.8	Experience Report (reflection)	Compares at least two different prioritization techniques with respect to relevant criteria.				
p.9	Experience Report (reflection)	Critically discusses experiences with prioritization.				

Criteria 1/5 (Specification)

Project assessment

ID	*Where checked*	*Criteria*	*F*	*3*	*4*	*5*
s.1	Requirements Document (content)	A sufficient number of suitable specification techniques has been applied (e.g. task descriptions, feature requirements, planguage).				
s.2	Requirements Document (content)	The specification covers goal, domain, product/design level requirements.				
s.3	Requirements Document (content)	Specification techniques are well combined across requirements types and abstraction levels.				
s.4	Requirements Document (content)	The level of detail and completeness is appropriate.				
s.5	Requirements Document (content)	The level of detail and completeness is well motivated.				
s.6	Requirements Document (scope)	The system boundaries and interactions with external entities are clearly defined.				
s.7	Requirements Document (scope)	Requirements include rationale to minimize the risk of misinterpretation.				
s.8	Requirements Document / Experience Report (scope)	Requirements balance the priorities of multiple stakeholders.				
s.9	Experience Report (reflection)	Critical reflection on specification trade-offs is provided.				
s.10	Requirements Document (dependencies)	Inter-dependencies between requirements are clearly identified and linked.				
s.11	Requirements Document (dependencies)	Important inter-dependencies are managed as requirements evolve, with a focus on cost-benefit considerations.				

Todo

→ `socrative.com`, Room REQENG, Q7

- Work in your groups: Meet with supervisor; finalize group setup
- This week:
  - Documentation Workshop
  - R1 review (cancelled)
- Next week: Traceability (Read [TRA])

References I

Deng, M., Wuyts, K., Scandariato, R., Preneel, B., and Joosen, W. (2011).
A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements.
Requirements Engineering, 16:3–32.

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

Glinz, M., Seyff, N., Bühne, S., Franch, X., and Lauenroth, K. (2023).
Towards a modern quality framework.
In *2023 IEEE 31st International Requirements Engineering Conference Workshops (REW)*, pages 357–361. IEEE.

Hosseini, M., Shahri, A., Phalp, K., and Ali, R. (2016).
Foundations for transparency requirements engineering.
In Daneva, M. and Pastor, O., editors, *Proc. of Requirements Engineering: Foundation for Software Quality (REFSQ)*, pages 225–231, Gothenburg, Sweden.

Houmb, S. H., Islam, S., Knauss, E., Jürjens, J., and Schneider, K. (2010).
Eliciting security requirements and tracing them to design: An integration of common criteria, heuristics, and umlsec.
Requirements Engineering Journal (REEN)), 15(1):63–93.

ISO/IEC25010 (2022).
Systems and software engineering - systems and software quality requirements and evaluation (square) - product quality model - part: Product quality model.
Technical report, ISO/IEC.
Draft International Standard.

References II

ISO/IEC25019 (2022).
Systems and software engineering - systems and software quality requirements and evaluation (square)- product quality model - quality-in-use model.
Technical report, ISO/IEC.
Draft International Standard.

Kasauli, R., Knauss, E., Kanagwa, B., Nilsson, A., and Calikli, G. (2018).
Safety-critical systems and agile development: A mapping study.
In *Proc. of Euromicro SEAA*.

Knauss, E., Liebel, G., Schneider, K., Horkoff, J., and Kasauli, R. (2017).
Quality requirements in agile as a knowledge management problem: More than just-in-time.
In *Proceedings of 2nd International Workshop on Just-In-Time Requirements Engineering: Dealing with Non-Functional Requirements in Agile Software Development (JITRE@RE17)*, Lisbon, Portugal.

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

Schneider, K., Knauss, E., Houmb, S., Islam, S., and Jürjens, J. (2012).
Enhancing security requirements engineering by organisational learning.
Requirements Engineering Journal (REEN), 17(1):35–56.
Special Issue on REFSQ 2011.

Svensson, R. B., Gorschek, T., Regnell, B., Torkar, R., Shahrokni, A., Feldt, R., and Aurum, A. (2011).
Prioritization of quality requirements: State of practice in eleven companies.
In *Proc. of 19th IEEE International Requirements Engineering Conference (RE)*, pages 69–78, Trento, Italy.

References III

Svensson, R. B., Sprockel, Y., Regnell, B., and Brinkkemper, S. (2012).
Setting quality targets for coming release with quper: an industrial case study.
Requirements Engineering Journal, 17:283–298.