Requirements Elicitation

Dr. Martin Hasitschka

Dashboard

SE&MB+G
pre-upload SE&M
with optional

SE4AIB+G
pre-upload SE4AI
with optional

SE&MB
pre-upload SE&M
without optional

SE4AIB
pre-upload SE4AI
without optional

SE&MI
presentation SE&M

SE4AI**B+**I
presentation SE4AI

SE&MB+G+O
post-upload SE&M

SE4AIB+G+O
post-upload SE4AI

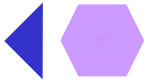
Big Picture: You have to ...

find

format

formulate

the requirements



Competence Map: At the End of this Course you will ...

1. The Importance of RE

...know the importance of RE for project success

2. Requirements Elicitation

... know how to elicit
basic requirements
performance requirements
excitement requirements
superfluous requirements

3. Requirements Structuring

... be able to structure

requirements documents

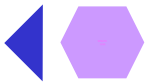
individual requirements

requirement backlogs

4. Requirements Documentation

… be able to write high-quality natural language requirements both for humans and AI systems

... be able to choose the right diagrams for requirements



The Top-10 RE Problems in Practice

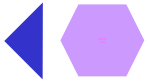
In a perfect world, we have **one** customer who provides us with high-quality requirements. In practice, however, we need to cope with questions like ...

1. How can we find **all** sources of requirements?
2. What can we do if the requirements sources disagree?
3. How can we elicit the requirements the customer has but doesn't tell us?
4. How can we find out if the customer **really** needs what she says?
5. What is the best prioritization method for requirements?
6. What is the best table of contents for a big requirements document?
7. What can we do to get **precise** natural language requirements?
8. What are the best diagram types for visualizing requirements?
9. What are the most important requirements attributes and relationships?
10. What can we do to keep the requirements up-to-date for years?

Requirements Sources and Stakeholders

Major requirements sources include
texts (e.g. laws, specifications, ...) – sometimes including diagrams,
systems (e.g. neighboring systems, legacy systems, competing products)
and
people that are influenced by the project or influence it („*stakeholders*“).

Without stakeholder identification, successful RE is not possible.



Self-Test: Stakeholders of «EarlyBird»

What stakeholders
does the online breakfast service
«EarlyBird» have?

Solution: Stakeholders of «EarlyBird»

- ✓ Customers (who want to order breakfast)
- ✓ Customers with disabilities
- ✓ Packaging clerks
- ✓ Delivery clerks
- ✓ Management of «EarlyBird» (esp. financial statistics)
- ✓ Procurement of «EarlyBird» (which products are successful?)
- ✓ Marketing of «EarlyBird» (which campaigns are successful)?
- ✓ Works council (no statistics for the performance evaluation of clerks)
- ✓ Data protection officer, as «EarlyBird» collects personal data
- ✓ Regulators (food regulations)
- ✓ Operations (operatbility requirements)
- ✓ Testers (testability requirements)
- ✓ Designers of neighboring systems, e.g. payment or texting

Class Group Exercise: Stakeholder Identification

Voltcar (VC) is a company developing self-driving cars. It has outsourced its software development to a separate company: VC-IT. VC-IT wants to replace the following four systems by a single new one:

- System **A**: Here, the customer (i.e. Voltcar) enters and tracks new requirements.
- System **B**: Here, VC-IT decomposes the requirement into tasks. The effort for each task is estimated separately with the help of AI. The total estimation for each task is fed back to System A so VC can see what the new requirement will cost and decide whether to order it.
- System **C**: Here, the completion of the tasks is planned and tracked.
- System **D**: Here, work effort for each task is logged.

Identify at least ten important stakeholders (by role, not name. Example: product owners of the four old systems). After five minutes, get help from an LLM.

What is Your Stakeholder Coverage?

Green: The stakeholder was successfully identified in the project.

Red: It was missed, which damaged the project.

- ✓ VC **users** (who enter requirements into A)
- ✓ VC-IT **requirements engineers/business analysts**
- ✓ VC-IT **developers**
- ✓ VC-IT **testers**
- ✓ VC-IT **project managers** for reporting
- ✓ VC and VC-IT **line managers** for reporting
- ✓ **operations** (which is why DevOps is a good movement)
- ✓ **legal.** Example: No one is allowed to log more than 12 hours per day.
- ✓ VC-IT **works council**
- ✓ **suppliers** because some tasks perhaps are outsourced.

Stakeholder Coverage – AI Version

Slidetext, but with 20 instead of 10 (ChatGPT, 11.9.24)
VC-IT Development Team – Responsible for coding and building the new system.
VC-IT Project Manager – Manages the project timeline, resources, and overall progress.
VC-IT System Architects – Design the overall architecture and technical solution for the new system.**VC-IT Test Engineers** – Handle quality assurance, ensuring the new system works as intended.**VC-IT Business Analysts** – Gather and analyze business requirements to ensure the new system meets Voltcar's needs.**VC-IT Support Team** – Provide support during the system transition and after the new system goes live.**Voltcar Management Team** – Oversee the investment and strategic alignment of the new system with company goals.**Voltcar End Users** – Those at Voltcar who will input and track requirements and tasks using the new system.**Voltcar Finance Team** – Ensure cost estimations and financial planning related to the new system are aligned.**Voltcar IT Governance and Compliance Team** – Ensure the new system meets internal policies and industry regulations. **VC-IT Security Team** – Ensures the new system is secure and complies with cybersecurity standards.**VC-IT Data Migration Specialists** – Handle the migration of data from the old systems to the new one.**VC-IT UX/UI Designers** – Responsible for designing a user-friendly interface for the new system.**VC Procurement Team** – Manage contracts, purchases, and vendor relationships for software tools or external services.**VC Legal Team** – Ensure that all legal requirements, including data privacy and contractual obligations, are met.**Third-Party Vendors or Consultants** – External experts who provide specialized knowledge or services for the system migration.**VC Operations Team** – Ensure that the new system aligns with operational processes and does not disrupt the workflow.**VC-IT Change Management Team** – Focus on managing the organizational change resulting from the new system implementation.**VC-IT Training Specialists** – Develop training programs for Voltcar staff to use the new system effectively.**VC Customer Support Team** – Ensure that customer-facing aspects of the new system, if any, operate smoothly and enhance the customer experience.

Missing: ***works council. But we didn't mention Austria.***

A More Precise Prompt

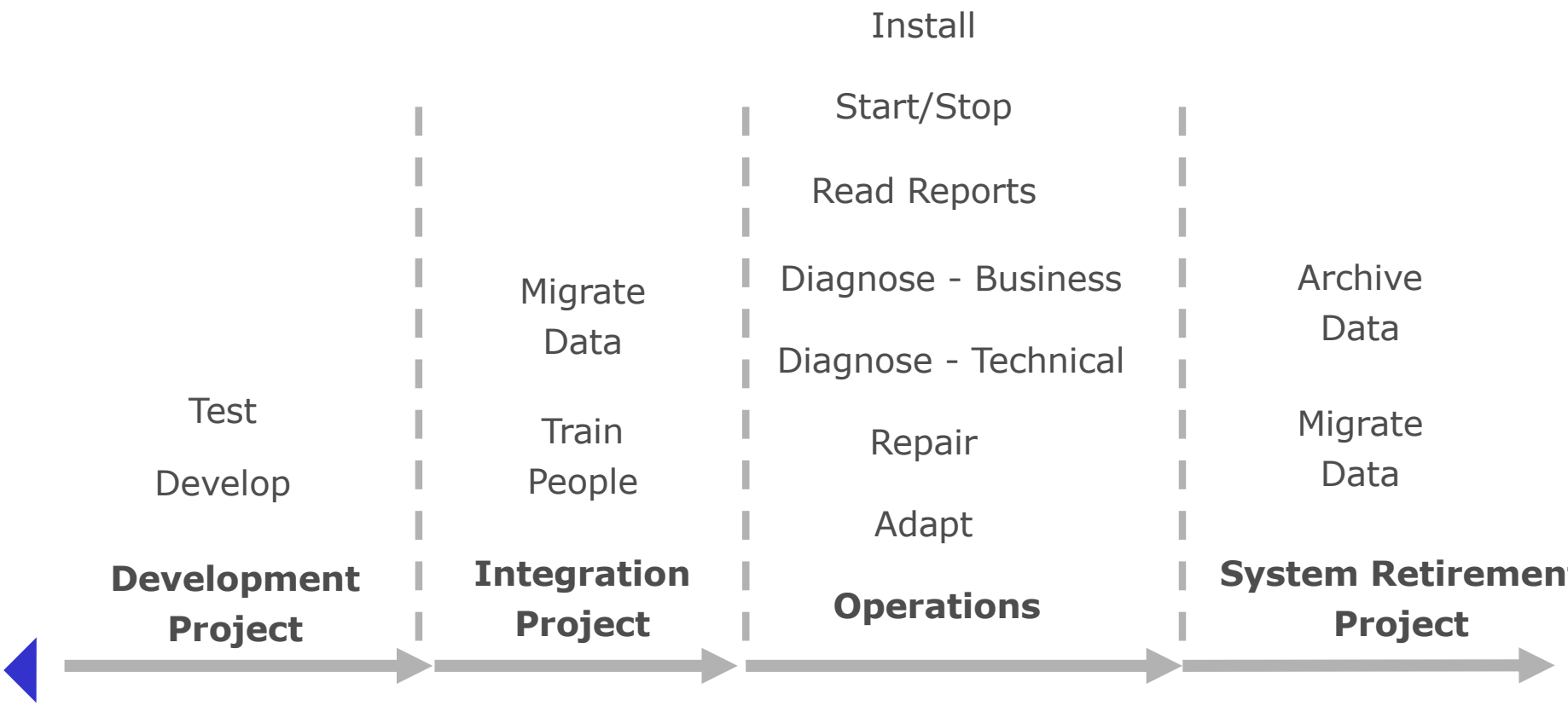
Given that this is a project in Austria, would you like to add stakeholders? (ChatGPT, 11.9.24)

Since this project is based in Austria, there are some additional local and region-specific stakeholders you might consider, especially in terms of regulatory, labor, and compliance aspects. Here are additional stakeholders relevant to an Austrian context:

1. **Austrian Data Protection Authority (DSB)** – Ensures the system complies with Austria's data privacy laws and GDPR (General Data Protection Regulation).
2. **Works Council (Betriebsrat)** – A legally mandated body in Austria representing employees' interests, particularly regarding the impact of new systems on jobs and working conditions.
3. **Austrian Labor Unions** – May be involved if the new system affects working conditions, job security, or labor rights.
4. **Austrian Regulatory Bodies** – Specific industry regulators or authorities overseeing compliance in the automotive or technology sectors.
5. **VC-IT Compliance Officers (Austrian/European Regulations)** – Ensure the new system adheres to Austrian and EU-specific industry standards and regulations.
6. **Austrian Software Licensing Authorities** – Verify that all software licensing in the new system complies with Austrian laws.
7. **Local Austrian Tax Authorities** – Involved if the new system impacts financial reporting or tax compliance.
8. **Environmental and Sustainability Officers** – Austria emphasizes environmental sustainability, and the project may involve ensuring the new system adheres to green IT standards.
9. **Local Austrian IT Service Providers** – If external services or partners are required, local providers familiar with Austrian market needs could be stakeholders.
10. **Public Transportation or Infrastructure Authorities** – Relevant if the new system affects automotive-related projects linked with public infrastructure or transportation.

Application Life Cycle Management and Stakeholder Identification

One approach to identify stakeholders is based on the software life cycle. In each life cycle phase, different activities need to be performed. Each of these activities will be performed by some stakeholder, who probably has relevant requirements.

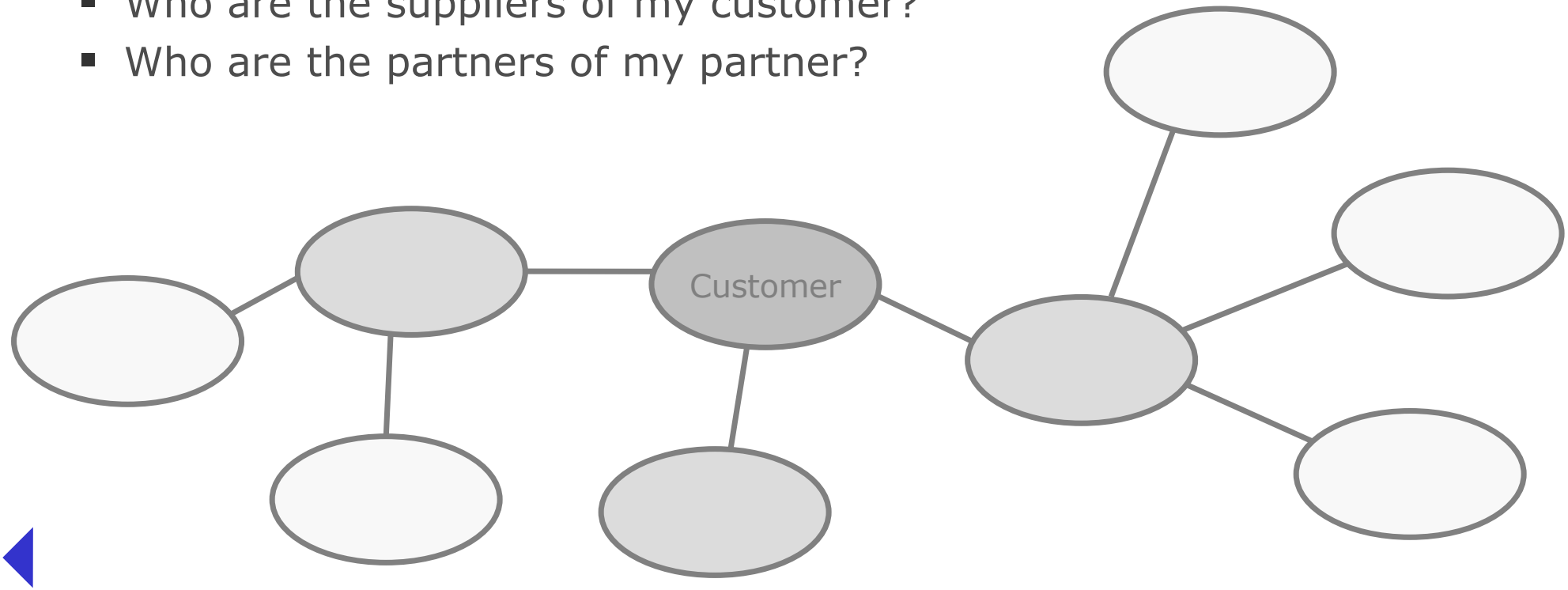


Stepwise Stakeholder Identification

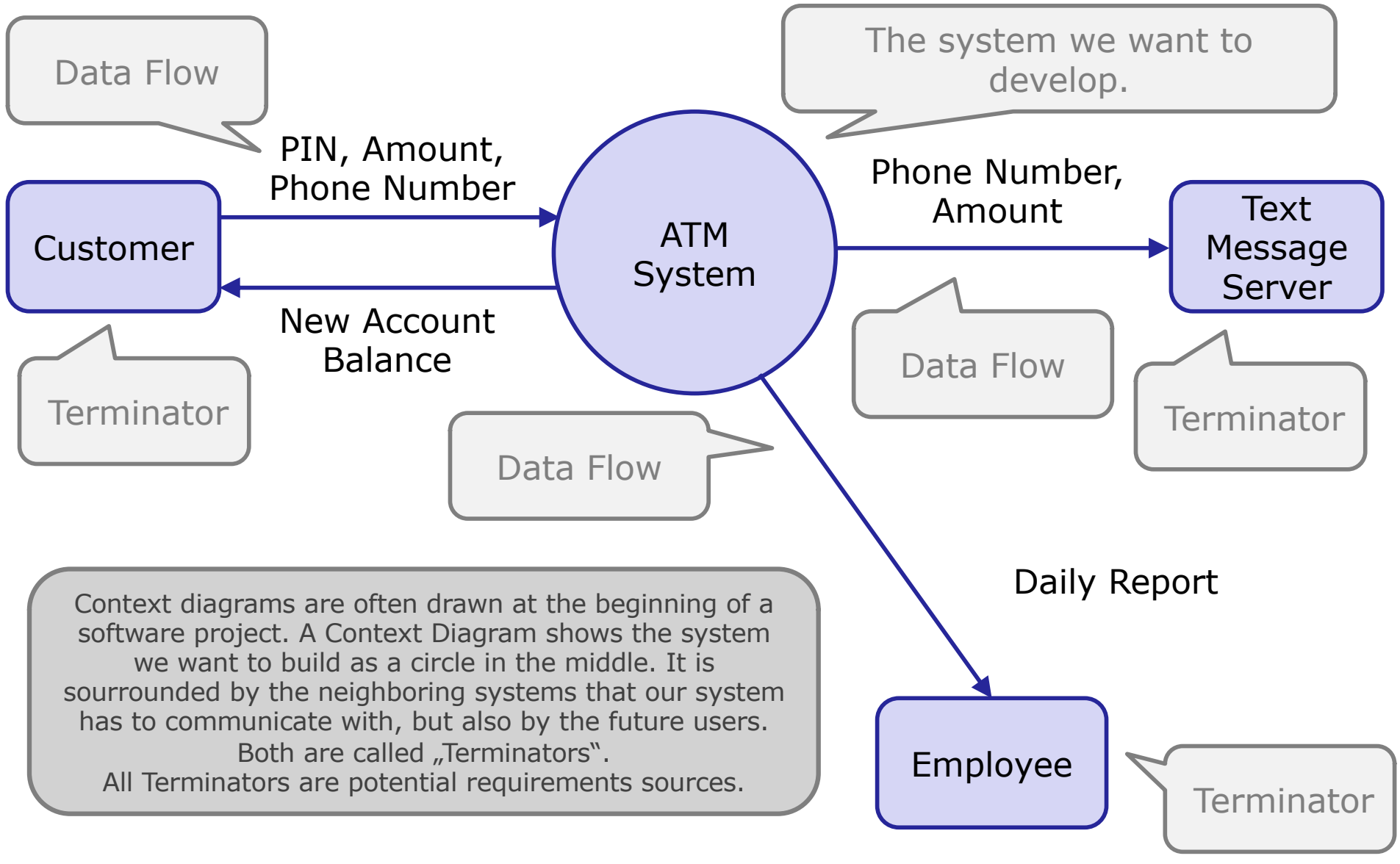
Stepwise stakeholder identification is based on the idea that if you have a stakeholder, her customers and/or suppliers and/or development partners perhaps are your stakeholders, too.

So you ask questions like

- Who are the customers of my customer?
- Who are the suppliers of my customer?
- Who are the partners of my partner?



A Context Diagram Shows Many Requirements Sources



Context diagrams are often drawn at the beginning of a software project. A Context Diagram shows the system we want to build as a circle in the middle. It is sourrounded by the neighboring systems that our system has to communicate with, but also by the future users. Both are called „Terminators".
All Terminators are potential requirements sources.

Stakeholder Documentation: Example

It is a recommended practice to have a stakeholder documentation, like in this example project.

	Barbara Knowles	Peter Nguyen
Role	IT-Architect	User
Contact	4578	3131
Availability	Mon, Thu 9 – 12 a.m.	Wed 2 – 4 p.m.
Knowledge	Certified architect, knows a lot about UML	Knows legacy system well
Goals	Simplification of the application landscape	As few differences to the legacy system as possible
Relevance	Decision maker	Information provider

Stakeholder Identification and Software Design

What do you do if two stakeholders have the same requirement? There are good arguments for behaving as if there were **two** distinct requirements, even though they appear to be the same.

Let's clarify this using an example from «EarlyBird»:
Req 155: There should be a list of all customers who ordered more than five breakfasts last year.

This requirement is wanted both by controlling and marketing. Robert C. Martin (a.k.a. "Uncle Bob") argues that in this case, you should implement two reports, not one, even though the look exactly the same (at first). "Robert C Martin - The Single Responsibility Principle", https://www.youtube.com/watch?v=Gt0M_OHKhQE, at 42:36
The reason is that in the future, the two reports are likely to evolve in different directions as the serve different stakeholders.

Stakeholder Identification and Software Design / 2

Uncle Bob talks about design here, but we also have to take a look at requirements engineering. It makes sense to split requirement 155 in two:

Req 155a: For controllers, there should be a list of all customers who ordered more than five breakfasts last year.

Req 155b: For marketing, there should be a list of all customers who ordered more than five breakfasts last year.

If you link source code to requirements in your project (which is an example of **requirements traceability**), you link the code of the first report to Req 155a and the code of the second report to Req 155b.

So, each requirement should refer to **one** stakeholder only.

Stakeholder Identification and Agility

How can we make sure that each requirement refers to **one** stakeholder only?

The user story format for wording requirements has brought a lot of progress here because it forces you to link each requirement to exactly **one** stakeholder:

*Req 155a: **As a controller**, I can display a list of all customers who ordered more than five breakfasts last year so that I can …*

*Req 155b: **As a marketing manager**, I can display a list of all customers who ordered more than five breakfasts last year so that I can …*

Class Group Exercise: Asking For Requirements

*„How do we elicit the requirements? Very simple: **We just ask**."*

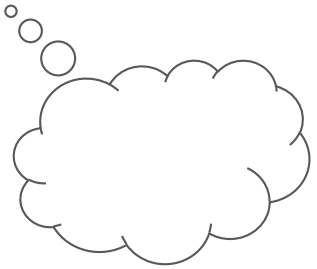
Tim Weilkiens, Systems Engineering mit SysML/UML, dpunkt, Heidelberg, 1. ed., 2006, p. 46

*„Most requirements are identified by **talking** to the various stakeholders about what problems they face in the status quo and what they need from the system to be developed."*

Christian Kästner, Machine Learning in Production: From Models to Products, The MIT Press, 2025

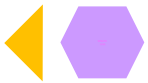
Is that really true? Let's try!

Case Study Alarm System



Requirements Coverage

1. ... be able to detect the opening of a window?
2. … be easily extensible if you add new windows to your house?
3. … have a hotline you can call if the alarm system has a defect - leading to a maintentance expert arriving at your house within four hours?
4. ... be able to detect motion in your house?
5. … be able to filter out motion caused by you, your family and your friends (using facial recognition software)?
6. … be able to detect negative events other than burglary, e.g. fire and flood?
7. … be able to activate itself automatically when no one is home and deactivate itself when a known face comes home?
8. … activate itself automatically when everyone in the house is asleep?
9. ... be able to call a predefined number in case of an alarm?
10. … be able to sound an alarm siren louder than 90dB (= hair dryer/lawn mower)?
11. … work also in case of a power outage (Stromausfall)?
12. …force you to renew its batteries only once a year or at even longer intervals?
13. ... send an alarm if someone tries to disconnect it from its energy source?
14. ... withstand jamming (jammer = Störsender) so that the communication between the sensors and the base station cannot be disrupted?
15. ... avoid interference with alarm systems in neighboring houses?
16. ... be able to send a photo in case of an alarm?



How Can I Get Sample Requirements to Measure Requirements Coverage?

Write 15 requirements for an alarm system of a house (ChatGPT, 11.9.24)

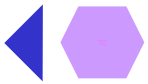
1. **Intrusion Detection**: The system must detect unauthorized entry through doors, windows, or other access points using motion sensors, contact sensors, or glass-break detectors. 2. **Alarm Activation**: When a breach is detected, the system must trigger an audible alarm to alert occupants and deter intruders. 3. **Remote Monitoring**: Users must be able to monitor and control the system remotely through a mobile app or web interface.4. **Emergency Alerts**: The system must send notifications (via SMS, email, or app) to the homeowner in case of an alarm trigger. 5. **24/7 Monitoring**: The system must integrate with a 24/7 professional monitoring service that can dispatch emergency responders if necessary.6. **Automatic Arming/Disarming**: The system must allow automatic scheduling for arming and disarming based on user preferences. 7. **Panic Button**: The system must include a panic button (physical or in-app) that immediately triggers the alarm and alerts authorities. 8. **Battery Backup**: The system must include a battery backup to ensure continued operation during power outages. 9. **Tamper Detection**: The system must detect any tampering with sensors or control panels and trigger an alert. 10. **User Access Control**: The system must allow multiple users to be added, each with customizable access levels (e.g., family members, guests) 11.**Reliability**: The system must maintain at least 99.9% uptime, ensuring that it is operational under most conditions.12. **Scalability**: The system must be scalable to add additional sensors or cameras as needed by the user. 13. **Data Encryption**: All data, including user information and notifications, must be encrypted both at rest and in transit to ensure privacy and security. 14.**Ease of Installation**: The system must be easy to install by the homeowner or come with professional installation options. 15. **Environmental Sensitivity**: The system must function in a variety of environmental conditions (temperature, humidity) typical for residential settings without false alarms.

The Alarm System Example and Requirements Elicitation

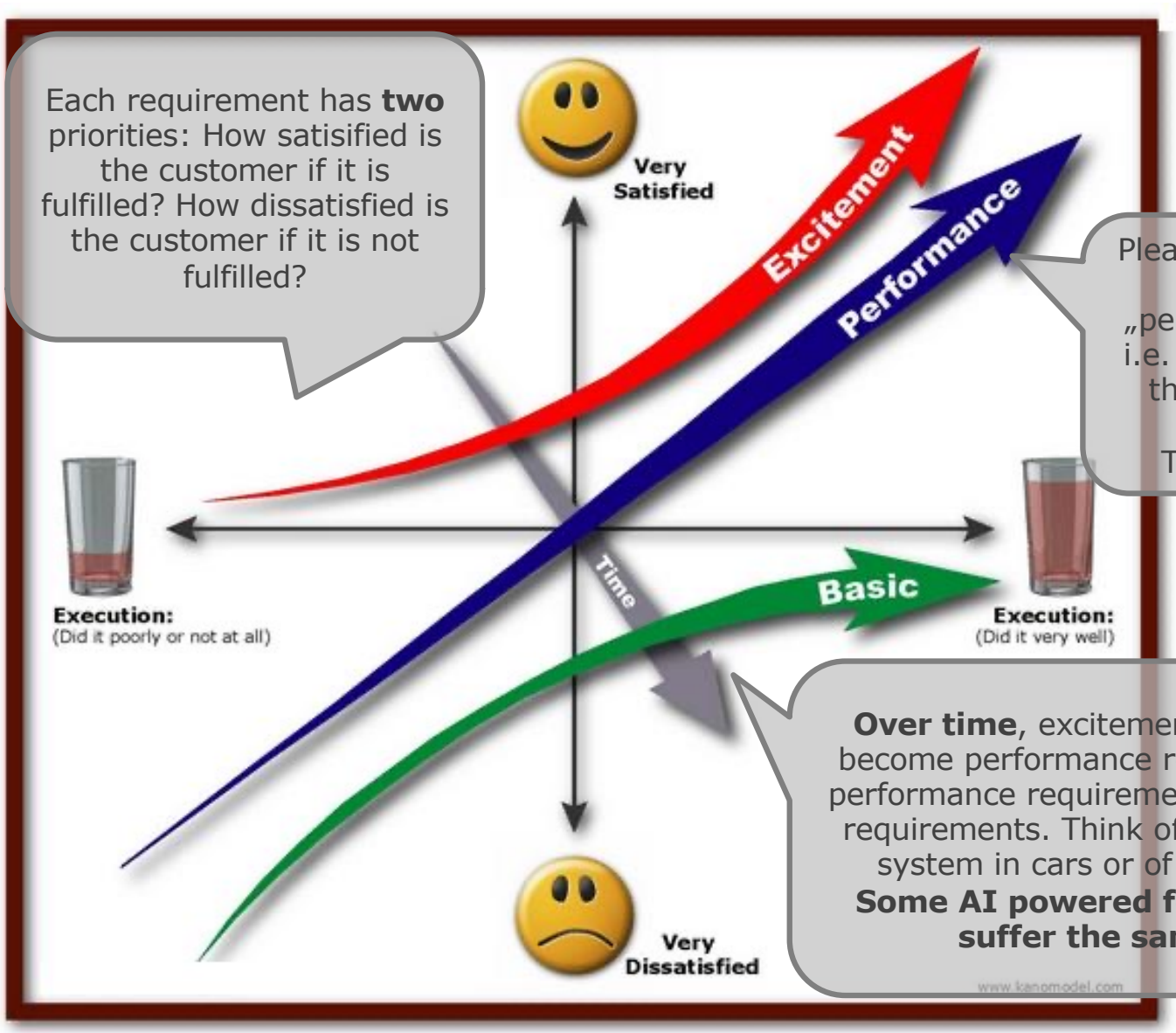
There were three types of requirements:

- Basic requirements (*Basisanforderungen*)
- Performance requirements (*Leistungsanforderungen*)
- Excitement requirements (*Begeisterungsanforderungen*)

They are not equally easy to elicit.



The Kano-Model

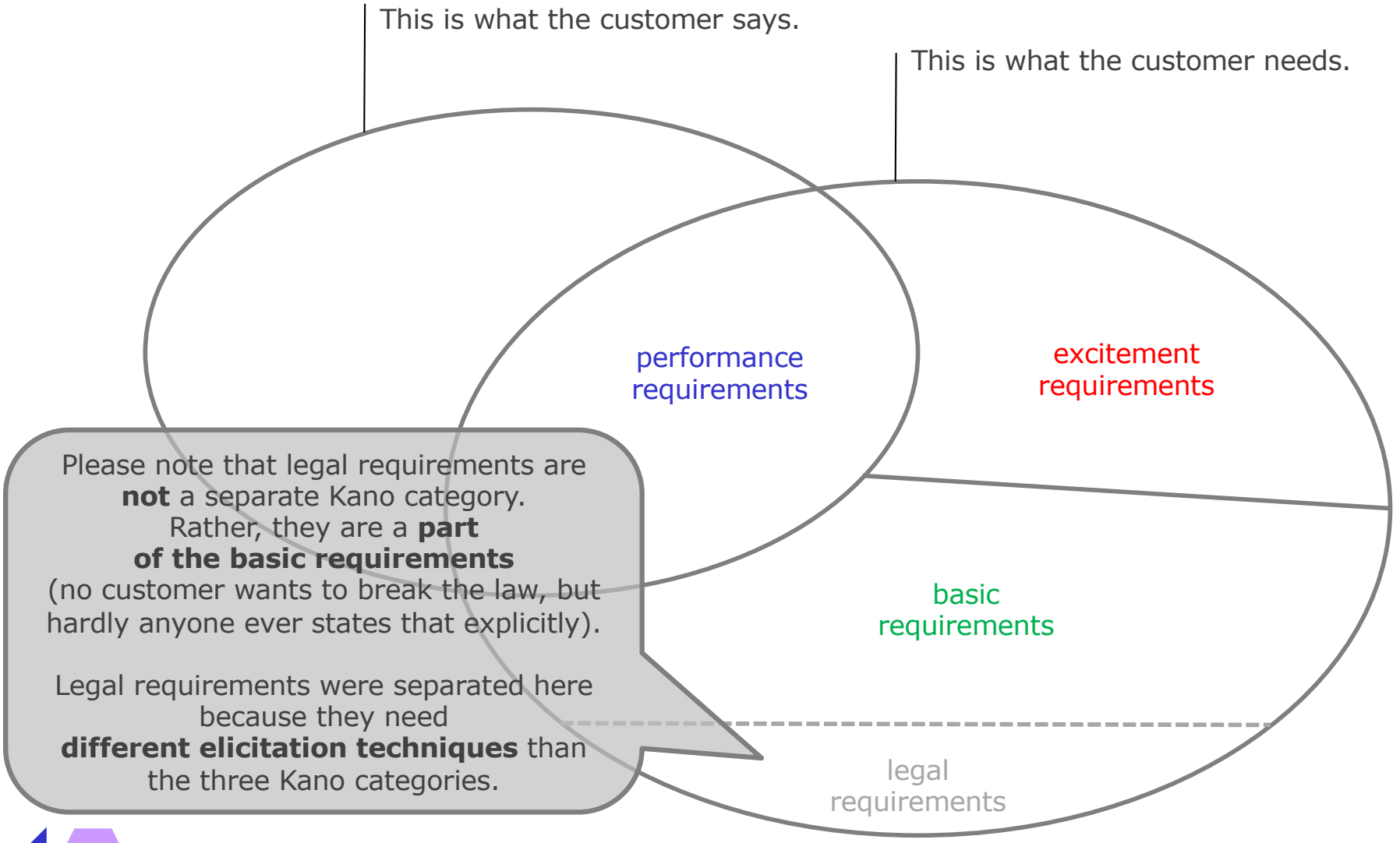


Each requirement has **two** priorities: How satisified is the customer if it is fulfilled? How dissatisfied is the customer if it is not fulfilled?

Please note that there is also another meaning of „performance requirement", i.e. a requirement placed on the usage of time and/or system resources.
This is not meant here.

Over time, excitement requirements become performance requirements and performance requirements become basic requirements. Think of the navigations system in cars or of smartphones.
Some AI powered features might suffer the same fate.

The Customer Only Tells us Performance Requirements



Kano and Requirements Elicitation

If you ask the customer „what do you want?“,
she will focus on performance requirements.
In other words:

Performance requirements are the only requirements type
that can best be elicited by **interviews**.

Basic requirements are elicited using different methods,
especially **prototyping** and **observation**.

Excitement requirements are elicited using yet different methods,
especially **creativity methods**.

Also bear in mind that there are stakeholders
whose requirements can only be elicited by **document analysis**
(esp. **legal** requirements, which is no Kano category).

The Top-10 RE Problems in Practice

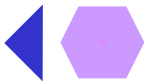
In a perfect world, we have **one** customer who provides us with high-quality requirements. In practice, however, we need to cope with questions like ...

Please note that these requirements typically are either basic requirements, legal requirements or excitement requirements.

1. How can we find **all** sources of requirem
2. What can we do if the requirements sou
3. How can we elicit the requirements the customer has but doesn't tell us?
4. How can we find out if the customer **really** needs what she says?
5. What is the best prioritization method for requirements?
6. What is the best table of contents for a big requirements document?
7. What can we do to get **precise** natural language requirements?
8. What are the best diagram types for visualizing requirements?
9. What are the most important requirements attributes and relationships?
10. What can we do to keep the requirements up-to-date for years?

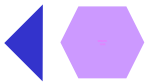
How can we Elicit the Various Requirements Types? Method Overview

Type	Skill	Method
Basic	Watch	Prototype, Agile Development, Observation
Legal	Read	Document Analysis
Excitement	Think	Creativity Methods
Performance	Talk	Interview

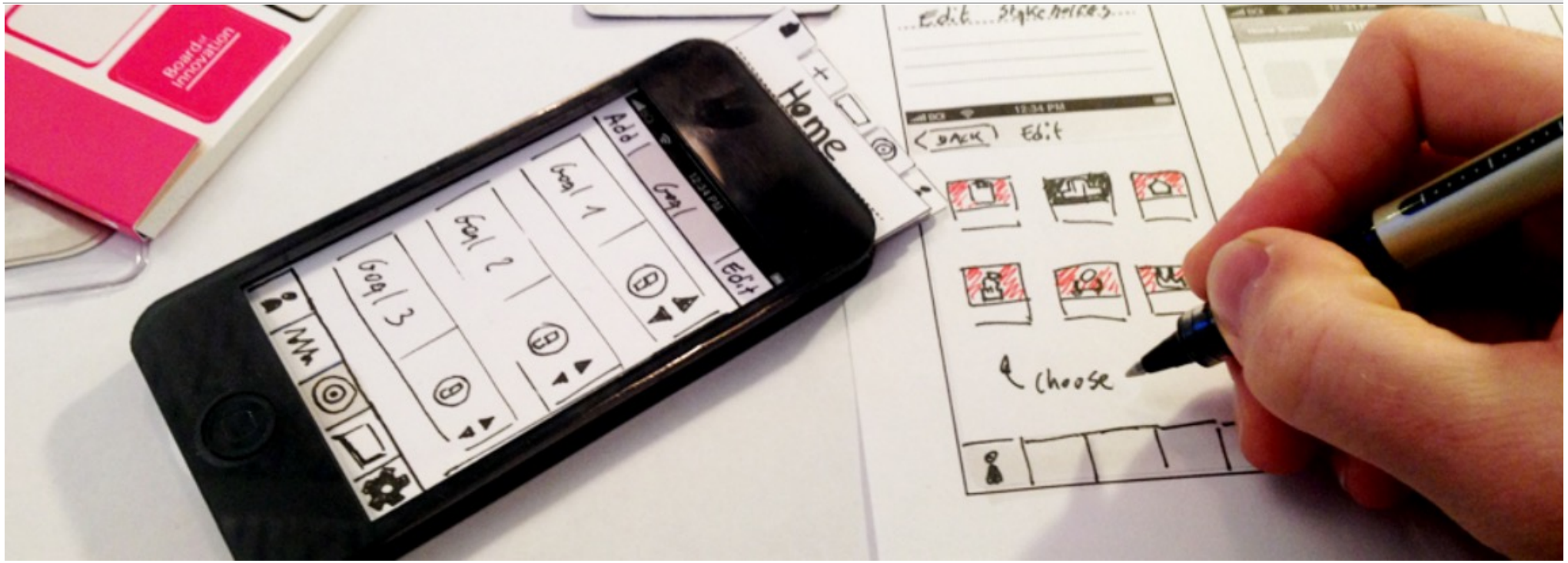


Elicitation Method Overview by Requirements Type

Type	Skill	Method
Basic	Watch	Prototype, Agile Development, Observation
Legal	Read	Document Analysis
Excitement	Think	Creativity Methods
Performance	Talk	Interview



Prototyping and Observation



A prototype is *"an individual that exhibits the essential features of a later type."* (Merriam Webster Dictionary)

In many cases, though, you don't build a prototype but simply watch the user handling the old system that should be replaced.

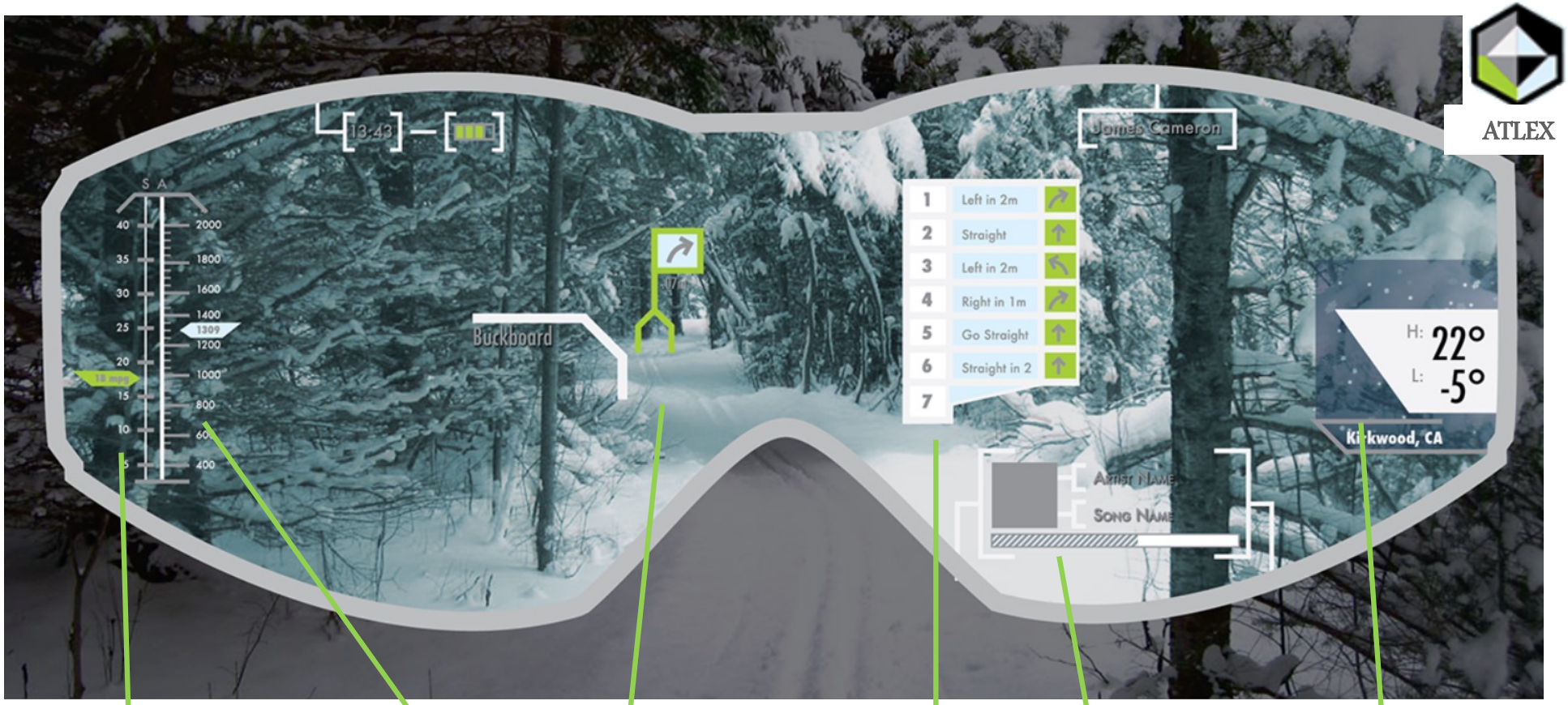
AI Makes Generating Software Prototypes Easy

Santiago
@svpino

Abonnieren

Let's be honest: AI has reduced the time and cost of building software prototypes to a point where it should be considered malpractice not to use it.

Example: GUI Prototype



Avalanche Risk

Snow Depth

Next Turn

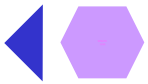
Next Turns

Music Info

Weather

Elicitation Method Overview by Requirements Type

Type	Skill	Method
Basic	Watch	Prototype, Agile Development, Observation
Legal	Read	Document Analysis
Excitement	Think	Creativity Methods
Performance	Talk	Interview



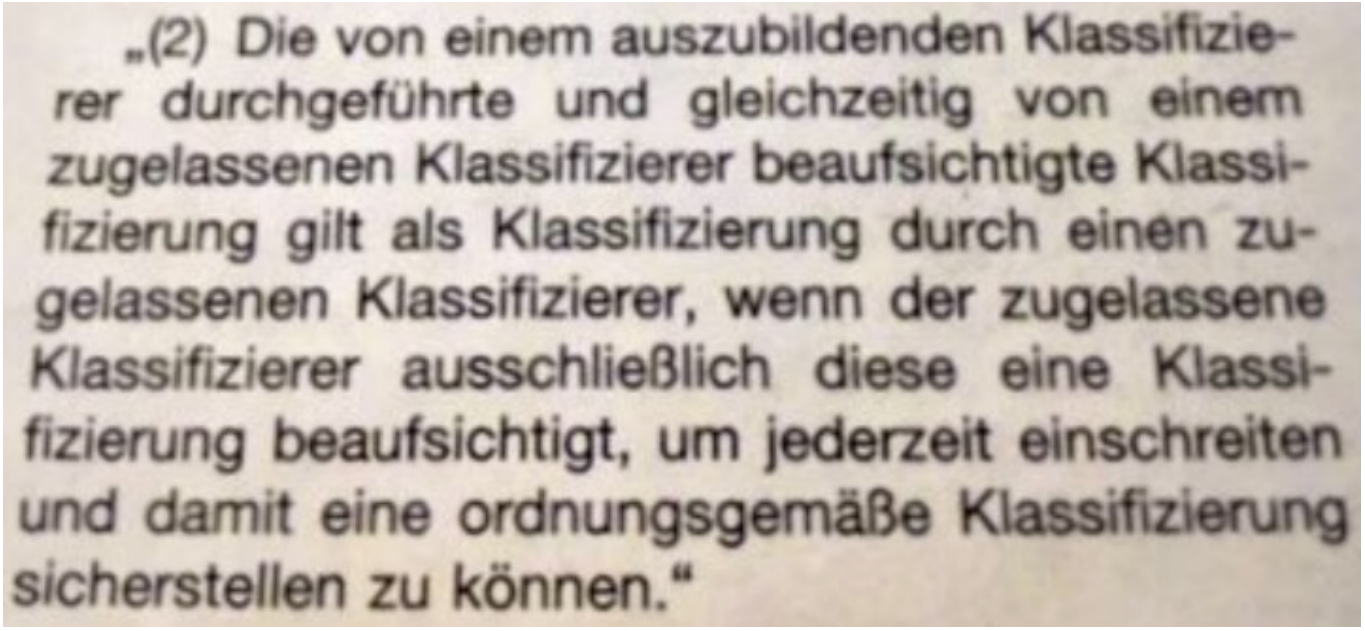
Legal Requirements are a Real Challenge

Three major problems with legal requirements:

1.) They are worded in an awkward language.

2.) You can't ask the stakeholder.

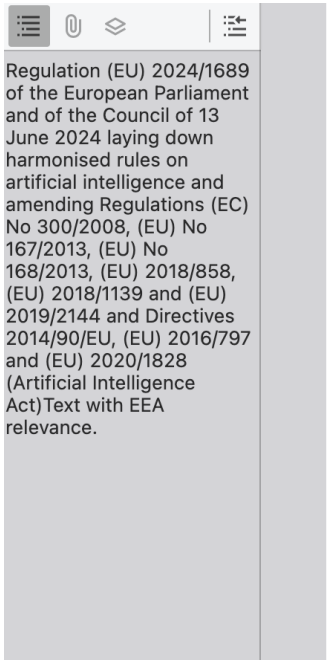
3.) The penalty for not fulfilling them is huge.



„(2) Die von einem auszubildenden Klassifizierer durchgeführte und gleichzeitig von einem zugelassenen Klassifizierer beaufsichtigte Klassifizierung gilt als Klassifizierung durch einen zugelassenen Klassifizierer, wenn der zugelassene Klassifizierer ausschließlich diese eine Klassifizierung beaufsichtigt, um jederzeit einschreiten und damit eine ordnungsgemäße Klassifizierung sicherstellen zu können."

Legal Requirements Example: EU AI Act

This is a small fraction of the 144 pages of the EU AI act as of 2024:

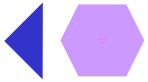


(24) If, and insofar as, AI systems are placed on the market, put into service, or used with or without modification of such systems for military, defence or national security purposes, those should be excluded from the scope of this Regulation regardless of which type of entity is carrying out those activities, such as whether it is a public or private entity. As regards military and defence purposes, such exclusion is justified both by Article 4(2) TEU and by the specificities of the Member States' and the common Union defence policy covered by Chapter 2 of Title V TEU that are subject to public international law, which is therefore the more appropriate legal framework for the regulation of AI systems in the context of the use of lethal force and other AI systems in the context of military and defence activities. As regards national security purposes, the exclusion is justified both by the fact that national security remains the sole responsibility of Member States in accordance with Article 4(2) TEU and by the specific nature and operational needs of national security activities and specific national rules applicable to those activities. Nonetheless, if an AI system developed, placed on the market, put into service or used for military, defence or national security purposes is used outside those temporarily or permanently for other purposes, for example, civilian or humanitarian purposes, law enforcement or public security purposes, such a system would fall within the scope of this Regulation. In that case, the entity using the AI system for other than military, defence or national security purposes should ensure the compliance of the AI system with this Regulation, unless the system is already compliant with this Regulation. AI systems placed on the market or put into service for an excluded purpose, namely military, defence or national security, and one or more non-excluded purposes, such as civilian purposes or law enforcement, fall within the scope of this Regulation and providers of those systems should ensure compliance with this Regulation. In those cases, the fact that an AI system may fall within the scope of this Regulation should not affect the possibility of entities carrying out national security, defence and military activities, regardless of the type of entity carrying out those activities, to use AI systems for national security, military and defence purposes, the use of which is excluded from the scope of this Regulation. An AI system placed on the market for civilian or law enforcement purposes which is used with or without modification for military, defence or national security purposes should not fall within the scope of this Regulation, regardless of the type of entity carrying out those activities.

(25) This Regulation should support innovation, should respect freedom of science, and should not undermine research and development activity. It is therefore necessary to exclude from its scope AI systems and models specifically developed and put into service for the sole purpose of scientific research and development. Moreover, it is necessary to ensure that this Regulation does not otherwise affect scientific research and development activity on AI systems or models prior to being placed on the market or put into service. As regards product-oriented research, testing and development activity regarding AI systems or models, the provisions of this Regulation should also not apply prior to those systems and models being put into service or placed on the market. That exclusion is without prejudice to the obligation to comply with this Regulation where an AI system falling into the scope of this Regulation is placed on the market or put into service as a result of such research and development activity and to the application of provisions on AI regulatory sandboxes and testing in real world conditions. Furthermore, without prejudice to the exclusion of AI systems specifically developed and put into service for the sole purpose of scientific research and development, any other AI system that may be used for the conduct of any research and development activity should remain subject to the provisions of this Regulation. In any event, any research and development activity should be carried out in accordance with recognised ethical and professional standards for scientific research and should be conducted in accordance with applicable Union law.

Elicitation Method Overview by Requirements Type

Type	Skill	Method
Basic	Watch	Prototype, Agile Development, Observation
Legal	Read	Document Analysis
Excitement	Think	Creativity Methods
Performance	Talk	Interview

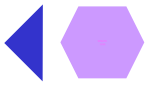


Creativity Method Overview

Most creative solutions are **not developed from scratch**.
Instead, they are based on other solutions that already exist.

But how can you get from already existing solutions to new ones?

- **Combination**: You take two (or more) existing solutions and combine them into a new one.
- **Variation**: You take an existing solution and change it to get a new one.
- **Inversion**: You take an existing **bad** solution and turn it into a new good one („*brainstorming paradox*“)

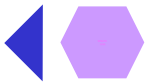


Creativity Method Overview

Most creative solutions are **not developed from scratch**.
Instead, they are based on other solutions that already exist.

But how can you get from already existing solutions to new ones?

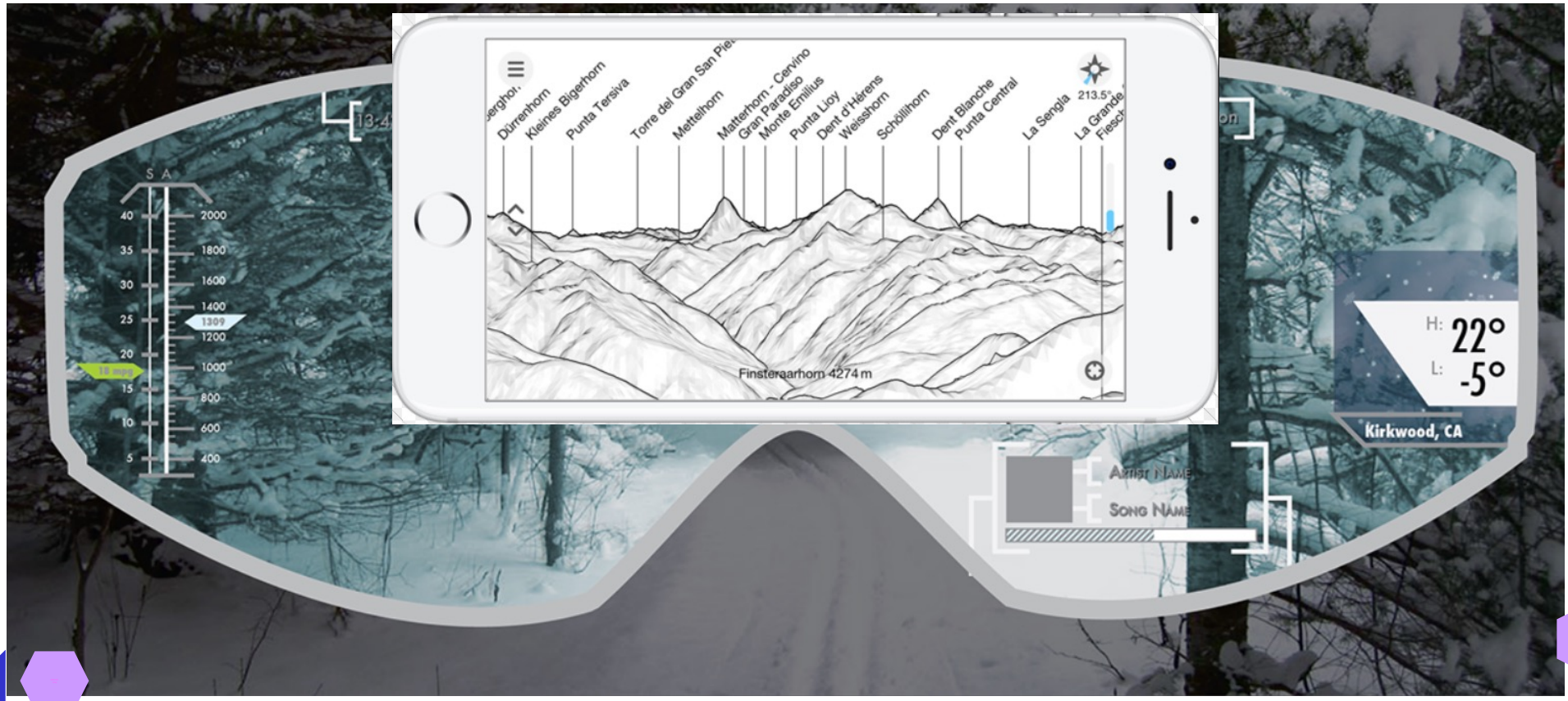
- **Combination**: You take two (or more) existing solutions and combine them into a new one.
- **Variation**: You take an existing solution and change it to get a new one.
- **Inversion**: You take an existing **bad** solution and turn it into a new good one („Brainstorming paradox“)



Example of the Combination of Existing Solutions

- Skiing goggles already exist.
- An app called PeakFinder that names the nearby mountains already exists.
- But the combination of the two is new: Goggles that show the names of the mountains nearby.

ATLEX

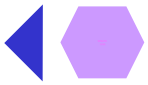


Creativity Method Overview

Most creative solutions are **not developed from scratch**.
Instead, they are based on other solutions that already exist.

But how can you get from already existing solutions to new ones?

- **Combination**: You take two (or more) existing solutions and combine them into a new one.
- **Variation**: You take an existing solution and change it to get a new one.
- **Inversion**: You take an existing **bad** solution and turn it into a new good one („Brainstorming paradox“)



Variation of Existing Solutions

Many creativity methods are based on changing an existing idea.

But what exactly should we change? Experience has shown that there are certain types of changes that are especially successful.

For mechanical engineering, there are well-known lists of those types of changes. An example is the Osborn-checklist. Here is a part of it:

- *Change colour, form or shape?*
- *Turn it backward, upside down or inside out?*
- *Split up? Combine?*

The Osborn-checklist is suitable for machines, but not perfect for IT-systems. For IT-systems, the perfect checklist is still waiting to be written.

An Osborn-Checklist for IT

Here is what could be the nucleus of an Osborn-checklist for IT systems:

- ✓ Add a neural network
- ✓ Interface to a Large Language Model
- ✓ Use eye tracking
- ✓ Communicate with similar products
- ✓ Use a compass
- ✓ Use an acceleration sensor
- ✓ Use GPS
- ✓ Split the screen
- ✓ Call a web service (navigation, weather, traffic, ...)
- ✓ Change the application domain (planes --> ships, cooking --> gardening, ...)
- ✓ Enhance the camera view with additional data (Augmented Reality)
- ✓ Allow the user to change the perspective (e.g. pilot view and ground view)
- ✓ Add an interface to social networks
- ✓ Use voice input
- ✓ Use voice output
- ✓ Count something and display the count
- ✓ Add a webshop
- ✓ ...

Example of the Variation of Existing Solutions

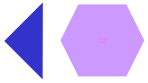
- Use eye tracking → The goggles warn you if you are tired.
- Use eye tracking → The goggles authenticate you via iris-scan and go dark if the result is negative.
- Change perspective (but also: Split the screen) → There is a back camera and the goggles show you in a small window what's behind you.
- Change the application domain → You also develop augmented reality **swimming** goggles.
- Communicate with similar products → The goggles show the position of your friends – if they have bought the goggles, too.
- Count something → The goggles show the number of kilometers you have been skiing today.

Creativity Method Overview

Most creative solutions are **not developed from scratch**.
Instead, they are based on other solutions that already exist.

But how can you get from already existing solutions to new ones?

- **Combination**: You take two (or more) existing solutions and combine them into a new one.
- **Variation**: You take an existing solution and change it to get a new one.
- **Inversion**: You take an existing **bad** solution and turn it into a new good one („*brainstorming paradox*“)



Example of Brainstorming Paradox

ATLEX

An example of very bad goggles are black ones that you can't see through.

This leads to the idea of night vision goggles for night skiing.

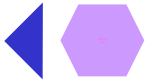
Home Exercise: Eliciting Excitement Requirements

Requirements Interviews

As we know from analyzing the Kano categories, if we interview the customer, she will tell us performance requirements only (or almost only).

Type	Skill	Method
Basic	Watch	Prototype, Agile Development, Observation
Legal	Read	Document Analysis
Excitement	Think	Creativity Methods
Performance	Talk	Interview

But that does not mean that everything we hear in such an interview is really needed by the customer. **How can we find those unnecessary requirements?**



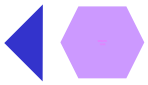
The Top-10 RE Problems in Practice

In a perfect world, we have **one** customer who provides us with high-quality requirements. In practice, however, we need to cope with questions like ...

1. How can we find **all** sources of requirements?
2. What can we do if the requirements sources disagree?
3. How can we elicit the requirements the customer has but doesn't tell us?
4. How can we find out if the customer **really** needs what she says?
5. What is the best prioritization method for requirements?
6. What is the best table of contents for a big requirements document?
7. What can we do to get **precise** natural language requirements?
8. What are the best diagram types for visualizing requirements?
9. What are the most important requirements attributes and relationships?
10. What can we do to keep the requirements up-to-date for years?

Requirements Rationale

Asking for the **requirements rationale**
(*„Why do you have this requirement?"*)
helps filtering out unnecessary requirements.



The Requirements Rationale

RATIONALE

The rationale should be kept up to date and include the following information:

- **Reason for the Requirement:** Often the reason for the requirement is not obvious, and it may be lost if

TCS

Requirement 2 (Priority: B)

At flight speeds of 1200 km/h or higher the thrust may not be increased, even if the pilot has expressed the desire to do so. **The reason is that the airplane has not designed for speeds beyond the sound barrier.**

Requirement 3 (Priority: B)

At flight speeds of 400 km/h or lower the thrust may not be decreased, even if the pilot has expressed the desire to do so. **The reason is that the airplane would stall.**

Source: https://www.nasa.gov/wp-content/uploads/2018/09/nasa_systems_engineering_handbook_0.pdf, 2025

The Library Project

A Requirements Interview

The requirements engineer is not just a passive recorder of requirements. Instead, she actively challenges the requirements, asking the customer for the reason for the requirement („*requirements rationale*“). **This leads to requirements that better suit the customers‘s needs.**

Librarian: „I want each customer with at least one overdue book to be marked with an asterisk on the customer list.".

Requirements Engineer: „Why with an asterisk?"

Librarian : „To recognize them faster."

Requirements Engineer : „Then maybe boldface would be better. Why do you want to recognize them?"

A Requirements Interview / 2

Librarian : „To write them faster to address labels."

Requirements Engineer : „Then some automated address label printing function would be better. But why do you need address labels?„

Librarian : „To send dunnings (German: *Mahnschreiben*)."

Requirements Engineer: „Then some automated sending of dunnings via e-mail would probably be better. But why do you want to dun at all?"

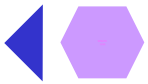
Librarian: „Because I want fewer books to be overdue.“

A Requirements Interview / 3

Requirements Engineer: „Then sending reminder e-mails in advance would probably be a good idea, too. But why do you want to have fewer overdue books?“

Librarian: „ To offer our customers more books for lending.“

Requirements Engineer: „You could try to increase the lending opportunities without increasing the number of physical books, e.g. by digitalizing the books thereby making each book available to more than one customer.“

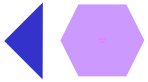


Three Important Questions to Ask about a Requirement

One of the most important questions
that the requirements engineer should put to the stakeholder:
Why do you have this requirement (**requirements rationale**)?

Another important question (not part of the example dialog)
is about **requirements priority**:
How important is this requirement to you?
Among other things, this helps backlog refinement and test planning.

Yet another important question (not part of the example dialog)
is about **requirements stability**:
How likely is it that this requirement will change in the future?
Among other things, this helps designing the software architecture.
Architecture = Framework for Change (Tom DeMarco)



Requirements Rationale and Agility

How can we make sure that each requirement has a rationale?
Again, the user story format for wording requirements
has brought a lot of progress here
because it forces you to describe a rationale
for each requirement:

Req 155a: As a controller, I can display a
a list of all customers who ordered more
than five breakfasts last year ***so that I can check whether we lost money making business with them.***

Req 155b: As a marketing manager, I can display a
a list of all customers who ordered more than five breakfasts last year
so that I can make them a more attractive offer.

Requirements Priority and Agility

How can we make sure that each requirement has
a defined priority?

Again, agility has brought a lot of progress here,
not through the story format
but because it introduced **ranked backlogs**. So

Req 155a: As a controller, I can display a
a list of all customers who ordered more
than five breakfasts last year so that I can …

can only be higher of lower than, but **not on a par with**

Req 155b: As a marketing manager, I can display a
a list of all customers who ordered more than five breakfasts last year so that I can …

Why Ask About Requirements Stability?

So we should ask whether a requirement is likely to change (which makes it an **unstable** requirement) or not (which makes it a **stable** requirement). But why?

Because they are **treated differently in architecture and design**. If a requirement is **unstable, you try to hide it behind an interface**. If the requirement changes later, you change the implementation, but not the interface.

This is why Robert C. Martin (a.k.a. Uncle Bob) says: *„[you need to figure out] where the **axes of change** really are, then you can implement the abstractions to protect us from the changes*“.
„FULL HOUR with Robert "Uncle Bob" Martin“,
https://www.youtube.com/watch?v=kScFczWbwRM, at 1:13:54.

Example: Requirements Stability and «EarlyBird»

Let's assume that «EarlyBird» has a certain requirement that describes how the transportation fee is calculated. The fee depends on the sum total of the products, the distance in km and the time of day:

Req 232: The transportation fee is 3% of the sum total of the products + 12 cents per km. Before 6 a.m., 15% are added.

You will probably hide the calculation of the fee behind an interface `ICalculateFee` (note that this very similar to the strategy pattern).

There are several reasons for doing so. One of them is that Req 232 is an unstable requirement. If the requirement changes, hopefully, you don't have to change the interface (and therefore the callers which may be many) but only the implementation.

Interview Language

The language of the requirements interview is that of the application domain (e.g. banking), not the solution domain (e.g. Java).
Avoid nerdy language!

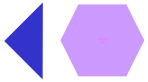
Good: *„Can a customer have more than one address?"*

Bad: *„What is the upper multiplicity bound of the customer – address association?"*

The Top-10 RE Problems in Practice

In a perfect world, we have **one** customer who provides us with high-quality requirements. In practice, however, we need to cope with questions like ...

1. How can we find **all** sources of requirements?
2. What can we do if the requirements sources disagree?
3. How can we elicit the requirements the customer has but doesn't tell us?
4. How can we find out if the customer **really** needs what she says?
5. What is the best prioritization method for requirements?
6. What is the best table of contents for a big requirements document?
7. What can we do to get **precise** natural language requirements?
8. What are the best diagram types for visualizing requirements?
9. What are the most important requirements attributes and relationships?
10. What can we do to keep the requirements up-to-date for years?



Conflict Resolution Techniques

Conflict resolution techniques include:

- **Voting** – the proposal getting the most votes is taken. Some requirements management tools support voting.
- **Escalation** – letting the superior (i.e. the boss) decide
- **Definition of variants** – e.g. a new system parameter so that everyone can have their proposal realized by choosing the proper value.
- **Version management** – Maintaining two versions of the product (e.g. a version for Europe and a version for the US)