

Workshop 1

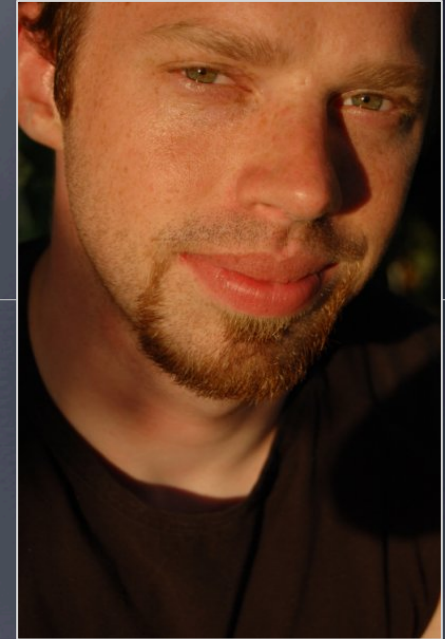
Requirements Specification, Natural Language Requirements and User Stories,
Quality of Requirements and SRS's

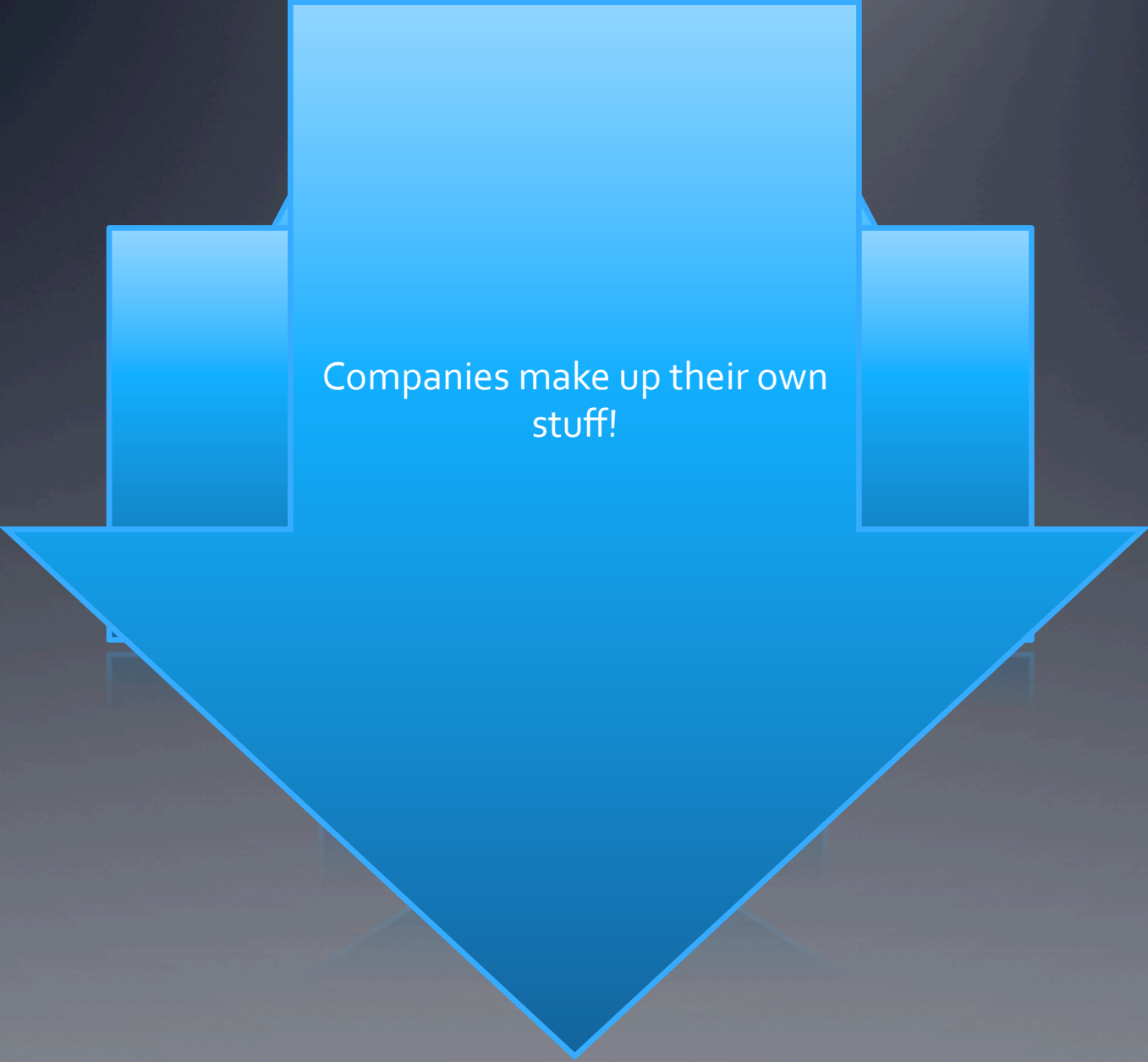
2012-09-07

Emil Alégroth

Who am I?

- Bachelor in Computer Science
- Master in Software Engineering
- PhD. Student in Software Engineering
 - Academic PhD. Student working close to industry
 - Focus area at the moment: Verification and Validation
- Contact:
 - Re.cse@chalmers.se (Course e-mail)
 - Jupiter building, 4th floor





Companies make up their own
stuff!

At a generic company

- You: Do you have Requirements?
 - Company: Reequireeements?
 - Company: AH! You mean documentation of customer needs and features...
 - Company: Aah, yes... No we don't have that.
 - You: ?????
 - Company: But we write stuff down on powerpoint slides in common language and then use that for design development etc.
 - You: So you have requirements?
 - Company: I wouldn't go that far as to call them that... *chuckle*
-

Dev team 1

Dev team 2

Dev team 3

Saab

Ericsson

What you will learn...

Requirements engineering

Requirements?

Customer Contact

Project Proposal

Requirements Elicitation

Requirements Specification

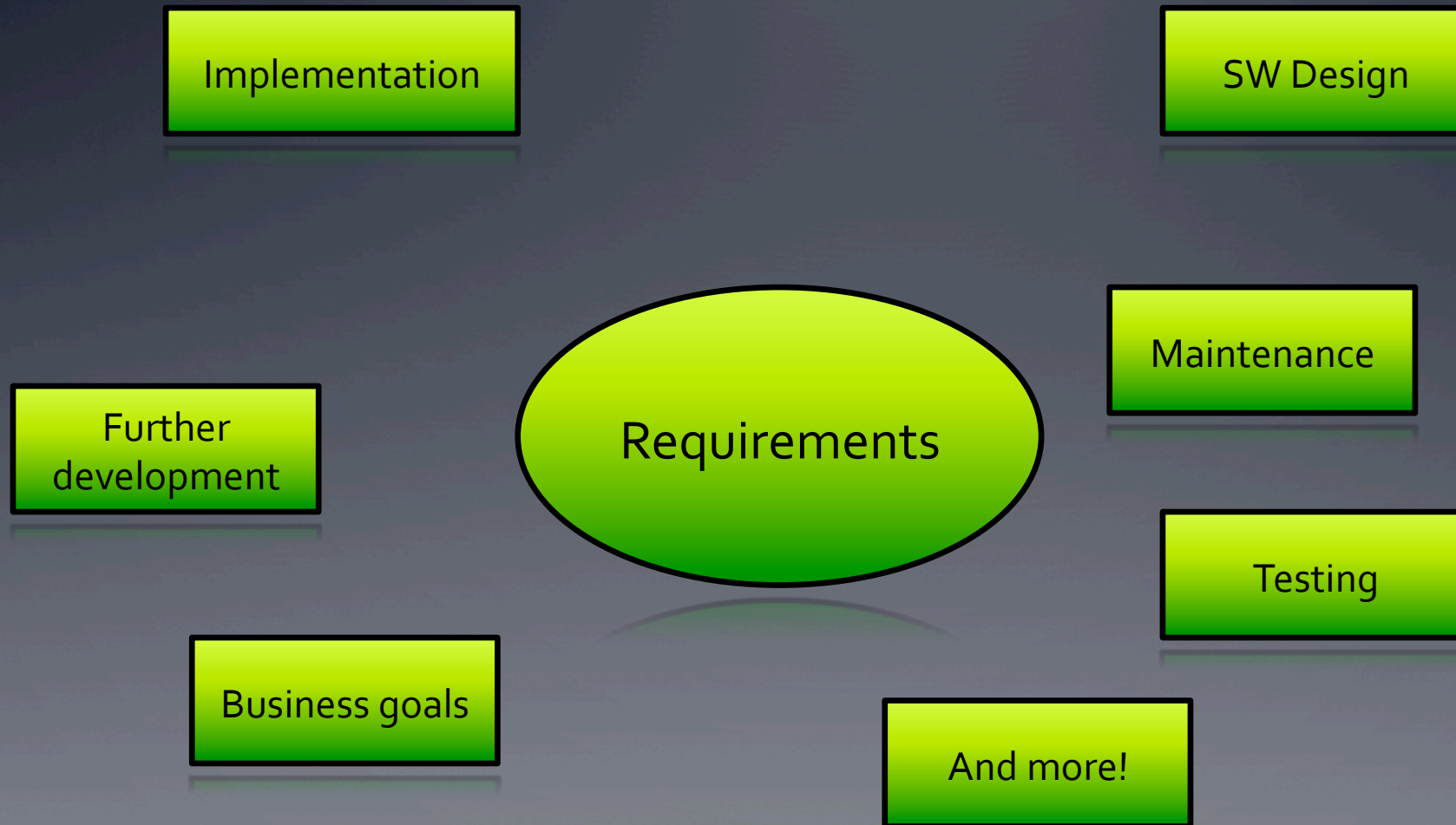
Design

Implementation

ETC

High-level
description of the
software
development
process.

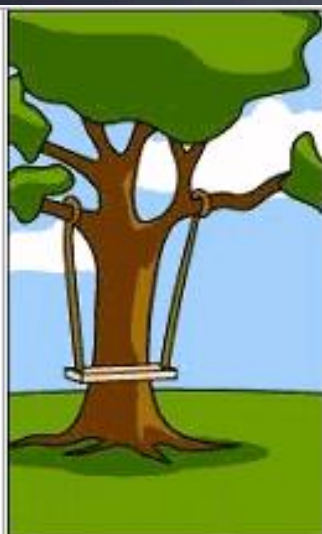
Overview



How hard can it be? ;)



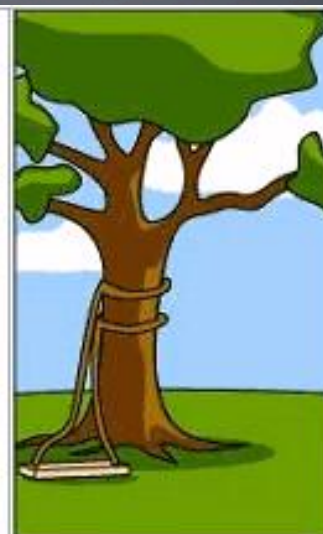
How the customer explained it



How the project leader understood it



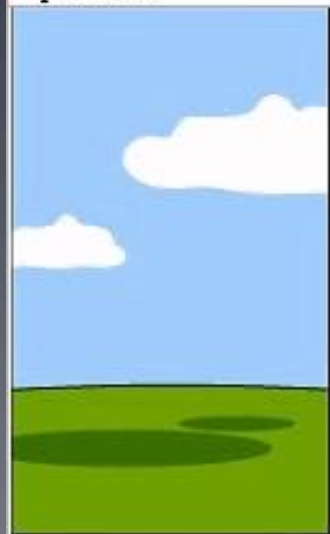
How the analyst designed it



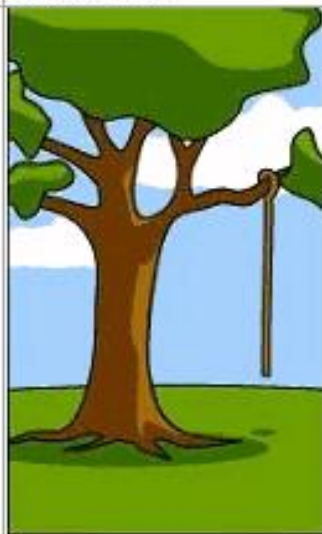
How the programmer wrote it



How the sales executive described it



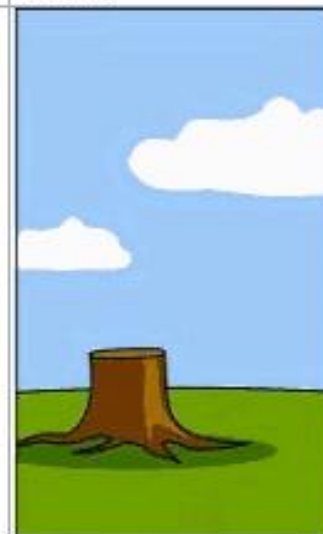
How the project was documented



What operations installed



How the customer was billed



How the helpdesk supported it



What the customer really needed

How hard can it be? ;)

- As it turns out... VERY!
 - Requirements are subject to a lot of problems
 - **Ambiguity** – Requirements can and will be interpreted differently by different people depending on their roles etc.
 - **Different taxonomies** – Different industrial domains, companies, even roles at a company have different understanding of different requirements.
 - **Work practices and processes** – Different processes (i.e. Agile vs Waterfall) handle Requirements engineering in different ways.
 - **Requirements change** – Customers don't know what they want, technology changes, competitors steal your idea, markets change, etc.
 - **AND MORE!!!**
-

How can we mitigate these problems?

- Requirements elicitation
- Requirements specification
- Multi-aspect and Multi-level specification
- Different methods



Requirements Specification?

“The deliberate documentation of requirements to a degree that makes the associated risks tolerable”

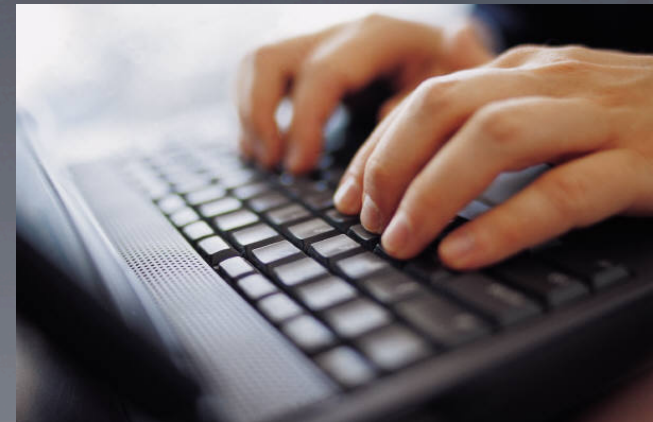
I.e. write down requirements in a specific way to avoid problems later during design, development, testing, etc.

Why?

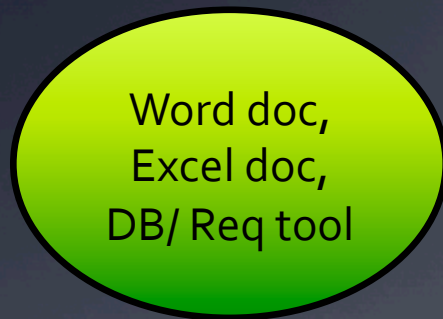
- Requirements are still ambiguous & open-ended after elicitation, which leads to:
 - Developers make decisions/assumptions later, which leads to:
 - **User <-> Dev difference:** User not satisfied
 - **Dev <-> Dev difference:** Inconsistent system
 - Overall: **Higher cost!**
 - BUT:
 - Goal is ideal PRODUCT not ideal Req Doc!
 - Thus: Just enough Req Spec to reduce Risks!
-

Purpose of Req. Specification?

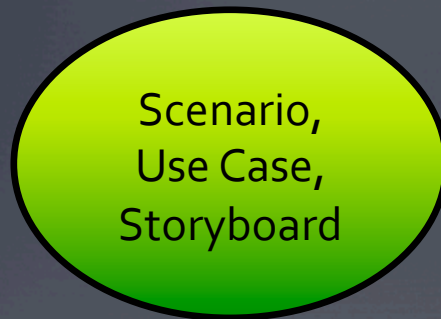
- Communication device between all parties
- Customers, Marketing, Sales, Finance, Management, Devs, Testers
- Drives design and choices
- Drives testing
- Drives project management
- Basis for evolution / releases



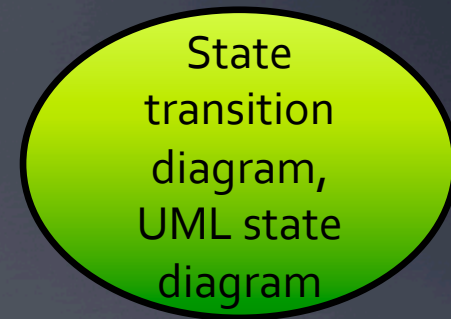
Specification Techniques



Text



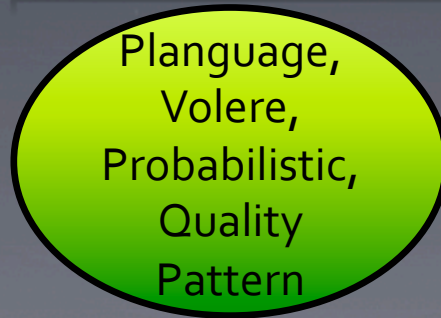
Interaction/
Sequence-based



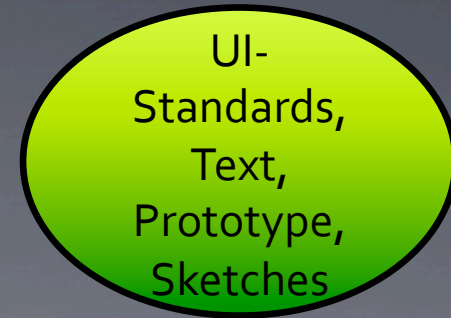
State-based



Decision-based



Quality
Requirements



User Interface



Formal

How do you document the requirements?

| Industrial Practice | Percentage Used |
|-------------------------------------|-----------------|
| Office (Word, Excel, Visio) | 23.8% |
| None | 15.3% |
| Requisite Pro | 10.2% |
| Quality Center | 9.6% |
| Don't know | 5.1% |
| Focal Point / DOORS | 4.0% |
| Caliber | 3.4% |
| Customer-Specific | 3.4% |
| RSA | 3.4% |
| Clear Case | 3.4% |
| Req Test | 3.4% |
| Rest/Other(max 2 mentions per tool) | 18.6% |

So much to choose from!?! ---

- Stakeholders must understand the requirements (Natural Language)
 - Models where NatLang has Limitations:
 - Complex interactions/sequences/states/decisions
 - Interfaces
 - BUT not “One model to rule them all!”
 - Quality requirements
 - Quantify
 - Capture in structured English or PLanguage
-



Natural Language Requirements

- What it sounds like, requirements documented in natural language sentences.
 - The exit button shall be red.
 - Structured and unstructured
 - Documented in Word, excel, powerpoint...
 - Easy for the customer to read
 - Ambiguous!
-

Aspects to think about!

- Use complete sentences! Use correct grammar & spelling!
- Keep sentences short
- Use Active Voice
- User Terms Consistently
- State requirements in a consistent fashion
 - ex: **The [actor] shall [action verb] [observable result]**
 - “The door management system shall display all users that have exited the building in the past 48 hours”
- Avoid Vague Terms. Avoid Comparative Words.
- RFC 211



RFC 2119?

- **MUST = REQUIRED = SHALL**
 - Absolute requirement of a specification
 - MUST NOT / SHALL NOT: Absolute prohibition
 - **SHOULD = RECOMMENDED**
 - May exist valid reasons to ignore in particular circumstances, but the full implications must be understood
 - SHOULD NOT / NOT RECOMMENDED
 - **MAY = OPTIONAL**
 - item is truly optional
-

Natlang requirement examples

- The quit button may be big.
 - The logoff button should be red and big.
 - The logoff button shall be red, located in the top left corner and be big.
 - The logoff button must be red, located in the top left corner of the UI and be 25% bigger than all other buttons.
 - The logoff button shall be red, located in the top left corner of the UI and be 500x500 pixels.
-

Structured NatLang (Example)

- ID number
 - Title
 - Preconditions
 - Postconditions
 - Rationale
 - Description
 - Relations
-

Structured NatLang Examples

Idnum: 5

Title: Save document

Preconditions: A document exists with unsaved data.

Postconditions: A new file/old file has been created/overwritten with unsaved data.

Rationale: Users want to save their work for the future.

Description: By clicking the save icon or selecting "save" from menu "Archive" the user can save the document.

Relations: None

Idnum: 6

Title: Close application

Preconditions: The application is running.

Postconditions: The application closes and data is saved.

Rationale: Users want to quit the application

Description: By clicking on the close button or selecting "Quit" from menu archive the user can terminate the application. If unsaved data exists in the document in the application the user is prompted with a save window.

Relations: 5

Use Cases

- Scenario based
 - Captures customer interaction and needs
 - Different types
 - Structured
 - UML support
-

Standard Use cases (IREB)

Designation

Name

Authors

Priority

Criticality

Source

Person responsible

Description

Trigger event

Actors

Pre-conditions

Post-conditions

Result

Main scenario

Alternative scenarios

Exception scenarios

Qualities

Designation: UC-0001

Name: Start computer

Description: User starts computer

Pre-condition: Computer is turned off

Post-condition: Computer is turned on

Main Scenarion:

1. User pushes "ON" button
2. Computer loads operating system (OS = Windows)
3. OS checks for updates
4. OS finds no updates
5. OS shows login screen to user
6. User enters password
7. OS checks password
8. Etc.

Alternative Scenario:

- 2B. Computer fails to load operating system
- 2C. Computer shows bluescreen of death
- 2D. Computer is not turned on

Use cases, another notation

Enter Building

Description: A user enters the building

Precondition: The person is a user in the system

Postcondition: Person has entered the building

User Intention

System Response

1. User swipes magnetic card

2. Verifies that card is valid

3. Asks for user code

4. User enters the code

5. Verifies that code is valid for swiped card.

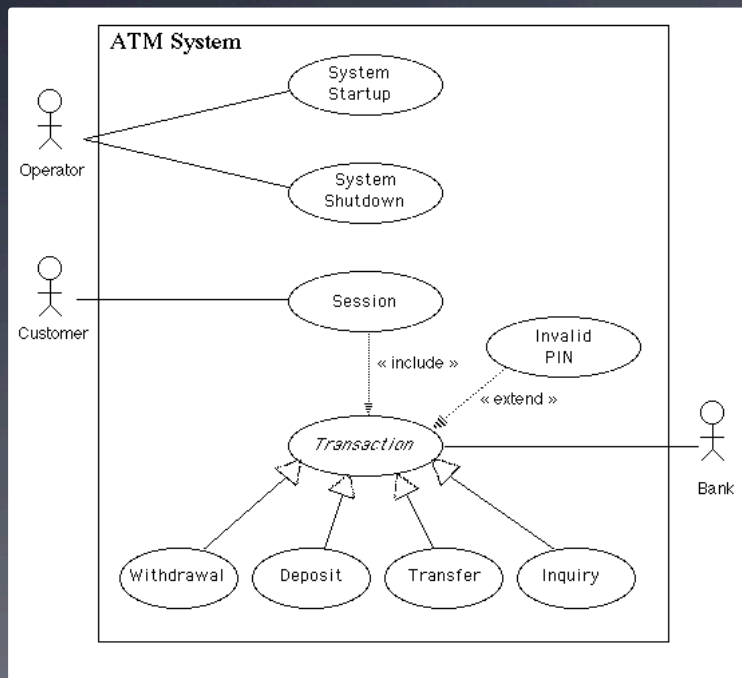
6. Opens door

7. Sound buzzer

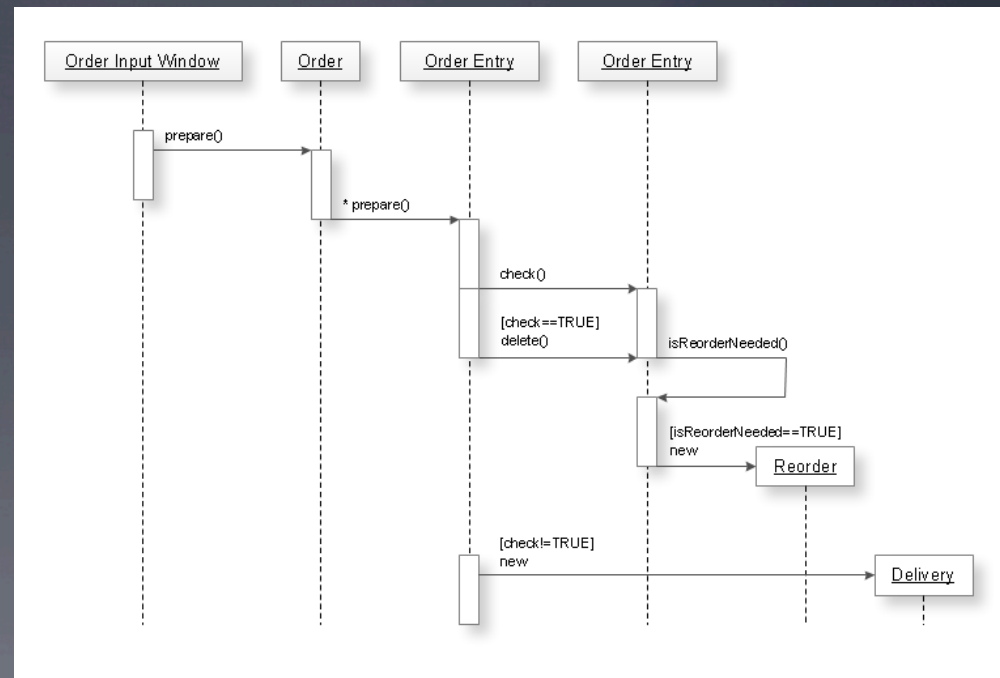
8. User opens door & enters building

9. Logs entry of user

Graphical Use Cases



Use case



Sequence Diagram

User Stories

- Written with customer on small notes (Agile practice)
 - Should be short and descriptive
 - Not finite once written, i.e. customer can change his/her mind
 - User stories can follow the templates:

As a <role>, I want <goal/desire> so that <benefit>
 - Or simplified as:

As a <role>, I want <goal/desire>
-

User Stories Examples

- Customer DB search
 - As a user, I want to search for my customers by their first and last name.
 - Drone Navigation
 - As an autonomous drone, I want to control my onboard navigation system.
 - Closing application
 - Upon closing the application, the user is prompted to save (When Anything has changed in data since the last save!)
-

User story card



Quality of Requirements?

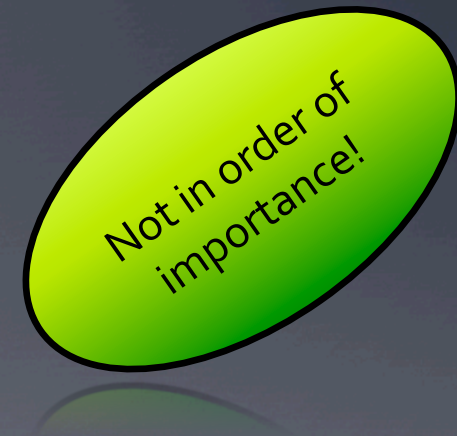
- How do we assess if a Requirement or a set of Requirements, i.e. a SRS, is of high quality?



- Quality attributes!
 - A set of aspects to consider when writing requirements, or,
 - A set of requirements, i.e. an SRS.
 - The attributes are not mutually exclusive!
-

Quality attributes?

- According to IREB, which we use in this course, the quality attributes are:
 - Correct
 - Unambiguous
 - Complete
 - Consistent
 - Ranked for importance and/or stability
 - Verifiable
 - Modifiable
 - Traceable
 - Clear structure
 - Agreed



Correct?

- A requirement is correct if it is valid, e.g. a property that should actually be in the system.
 - Examples
 - The text on the start screen shall read: "All your base are belong to us".
 - The sky shall be red.
 - A requirements specification is correct if every requirement in the specification should be in the system.
 - Example (Safety critical system)
 - The system shall warn the user if radar fails.
 - The system shall allow the user to play Solitaire.
-

Unambiguous?

- A requirement is unambiguous if it has only one interpretation.
- Example:
 - The start icon must look exactly like figure 1.
 - Figure 1:
- A requirements specification is unambiguous if all its requirements are unambiguous.
- Example:
 - Build a system, any system... derp!



(Impossible to exemplify!)

Complete?

- A requirement is complete if it completely describes all the functionality it specifies. Incomplete requirements should be marked as “to be determined” (tbd)
 - Ex:
 - The start button shall be red.
 - (tbd) The start button shall have a color.
 - A requirements specification is complete if it contains all the requirements a customer wants in the corresponding release.
 - Requirements are in constant flux
 - Customers have no clue what they want
-

Consistent?

- A requirement does not have this attribute!
- BAD example:
 - The start button must be blue and red... hmm... Purple?
- A requirements specification is consistent if none of the requirements conflict with another requirement, a set of requirements or a previously approved document.
- Obvious Example:

- The start button shall be red.
- The start button shall be blue.



Ranked for importance and/or stability?

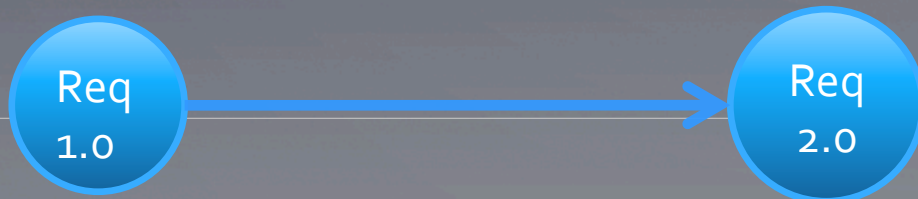
- A requirement is ranked if it includes a measure of its importance and/or stability.
 - Numbering system.
 - Req 1 > Req 2 > Req n
 - Necessity.
 - Essential, Conditional or Optional (Example!)
 - Stability (Volatility)
 - How many times is the requirements expected to change?
 - A requirements specification is ranked if all requirements have an identifier to indicate either stability or importance.
 - P – Priority, S – Stability
 - Example:
 - P_1, S_∞ , The system shall...
 - P_∞, S_1 , The system shall...
-

Verifiable?

- A requirement is verifiable if there exists some finite cost-effective way of testing that the system as built fulfills the requirement to a degree that satisfies all relevant parties.
 - Bad Example
 - The system shall **continuously** be controlled by **some** buttons.
 - How do we make a test for that? Ambiguity equals not verifiable.
 - A requirements specification is verifiable if all requirements in the specification are verifiable.
 - Better example (i.e. not good)
 - Output A shall be produced by the system 20 seconds after event X.
 - Testable, includes quantities. If A and X are also well defined everyone will live happily ever after.
-

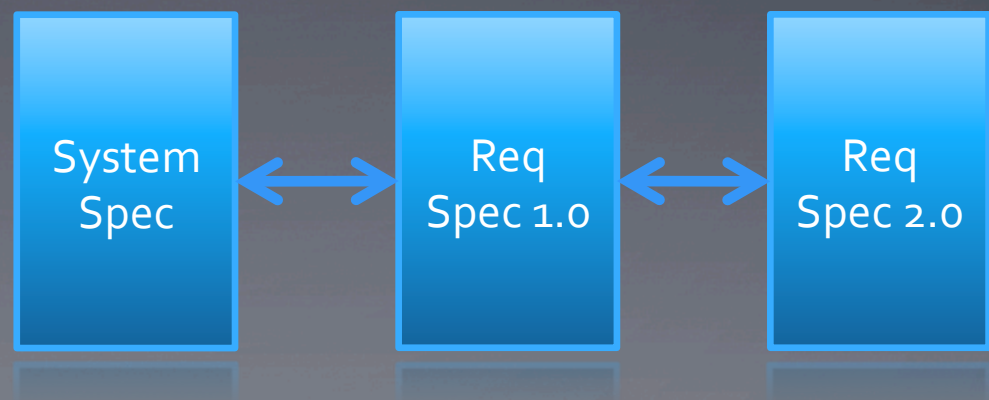
Modifiable?

- A requirement is modifiable if it can easily be modified without affecting the design or structure of the specification.
- Hence! Primarily a specification attribute!
- One template, e.g. **The [actor] shall [action verb] [observable result]**
- An SRS is modifiable if its structure and style are such that changes can be done easily to the requirements whilst keeping the completeness and consistency of the specification, its structure and style.
 - Coherence
 - No Redundancy
 - Separate requirements



Traceable?

- A requirement is traceable if it is backwards and forwards traceable.
 - Backward: Where did it come from? References to previous sources.
 - Forward: What spawned from the Req.? Ways of referencing the requirement.
- Example:
 - Ref: System req 3.2
 - Idnum: F-Req 5.6.2
 - Desc: The system shall...
- A requirements specification is traceable if the origin of each requirement is clear and it facilitates the referencing of each requirement in future development.



Clear Structure

- Single requirements do not have this attribute. But they themselves should be logically ordered on a meta-level to provide a clear structure.

Introduction

Functional Reqs.

Quality Reqs.

Models

Etc.

- An SRS should have a clear structure to be readable and comprehensive.
- Think about the order of requirements
- Ex.
 - ...
 - QR₁
 - FR₃₄
 - FR₃₅
 - QR₂
 - US₁
 - FR₃₆
 - ...

Agreed

- A requirement fulfills the agreed attribute if it is considered valid by all stakeholders.
 - All requirements in a SRS should be agreed for it to have high quality.
 - Ex of a not agreed upon requirement:
 - **SH1:** All buttons shall be red.
 - **SH2:** All buttons shall be blue.
-

(More..) Achievable/Feasible

- A requirement is feasible if it is implementable given current technology, resources, and delivery date.
 - **Examples:**
 - The system shall be able to send data to the other side of the world instantly.
 - The AI must have a neurologic pattern capable of rewriting its internal matrix at a capacity of 4 TB of data per second.
 - A requirements specification is achievable if it is possible to construct a system including all the requirements from that specification.
 - **Example:** One week before deadline, customer changes their mind that response-time of the system should be 4 ms instead of 40 and that ALL the UI components must be changed...
 - What do you do?
 1. Work 25 hours a day?
 2. Tell the customer to go BEEEEEP themselves?
 3. Renegotiate the contract
-

Annotated

- A requirement is annotated if it is easy to find characteristics of the requirement, including its relationships with other requirements.
 - Origin of a requirement.
 - Example:
 - Customer: "I want ALL buttons to be red, it's my favorite color you know."
 - Requirement:
 - Id: 12315215
 - Desc: Blablblablaaa...
 - **Rationale**: The button should be red because it's the customers favorite color.
-

Assignment 1

- **Assignment 1A: AtmosFear** - yymmd1d2. if d2 is even (ex: 1, 3, 15, 25)
 - **Assignment 1B: eVac** - yymmd1d2. if d2 is odd (ex: 2, 8, 18, 24)
 - 20 NatLang (10 good/10 bad)
 - Both Functional and Non-Functional requirements
 - 5 use cases
 - 3 user stories
 - Software requirements!
 - SurveyMonkey (<https://www.surveymonkey.com/s/re-2012-srs-review>)
 - Deadline
 - **Submit through FIRE, 13/9 at 18.00!**
-

Questions?
