Workshop 2

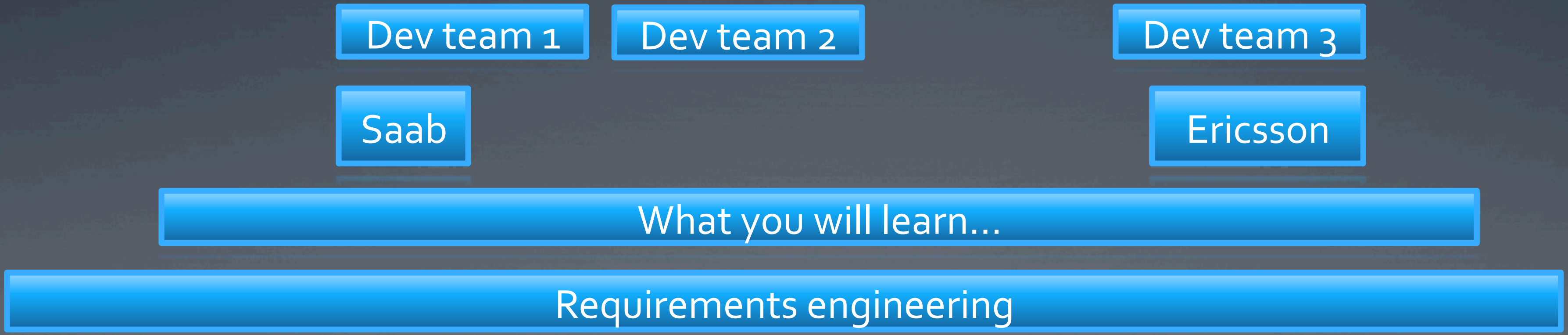
Quality of Requirements and SRS:s

Emil Börjesson 2011
Emil.Borjesson@Chalmers.se

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

From last week.

- Write in active voice:
  - Preferably Shall not Should. I.e. The button shall be red.
- Domain dependent!
- Be consistent, important for readability.

Quality of Requirements?

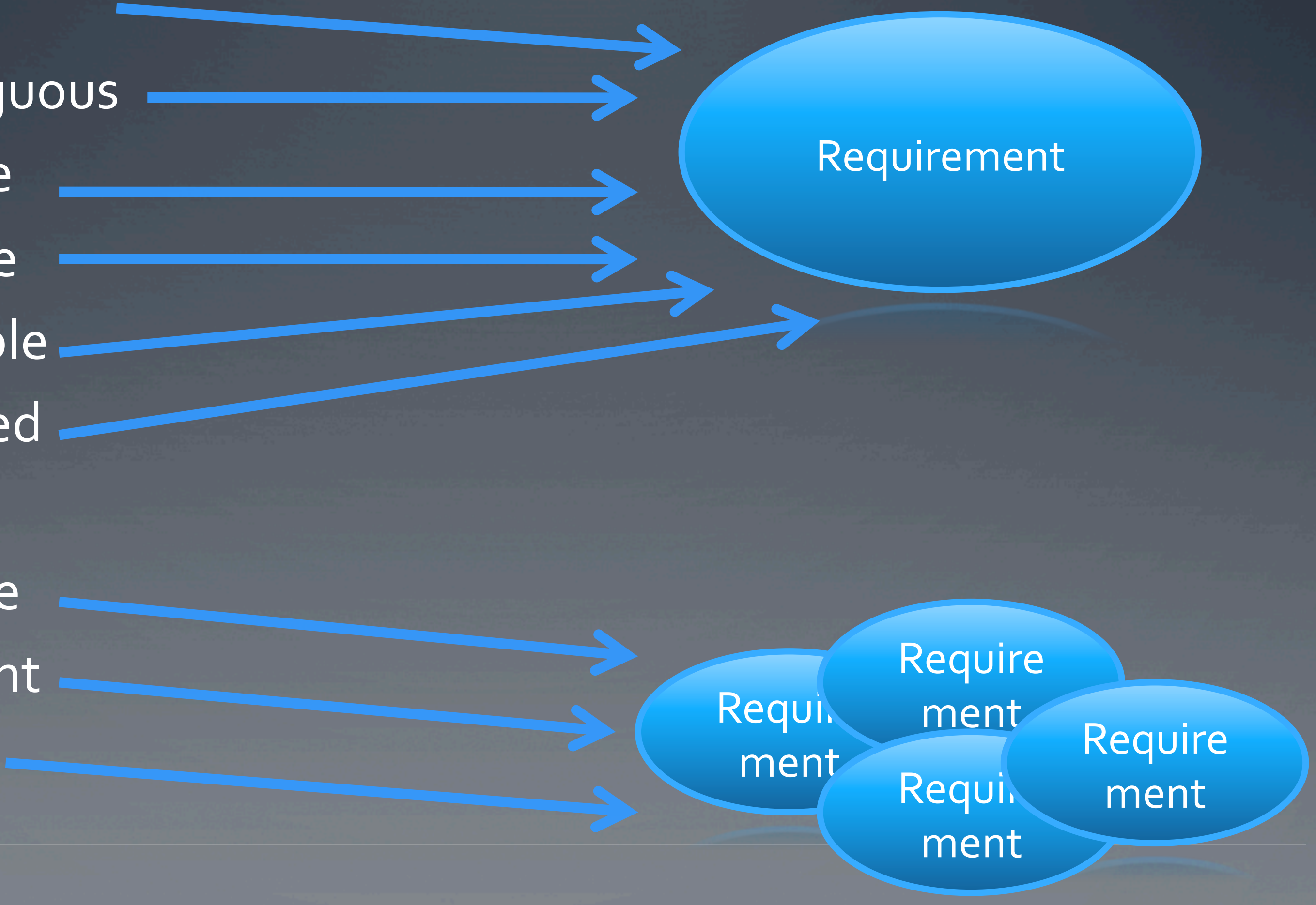
- How do we assess if a Requirement or a set of Requirements, i.e. a SRS, is of high quality?

- 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! Davis book is a bit dangerous.

Which are the Quality aspects for a Requirement?

- According to Davis book they are:
  - Correct
  - Unambiguous
  - Verifiable
  - Traceable
  - Achievable
  - Annotated

  - Complete
  - Consistent
  - Other



Which are the aspects for an SRS?

- According to the IEEE 830 standard that we will use in this course, the quality attributes of an SRS are:
  - Correct
  - Unambiguous
  - Complete
  - Consistent
  - Ranked for importance and/or stability
  - Verifiable
  - Modifiable
  - Traceable

Not in order of importance!

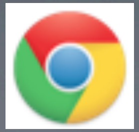
Correct?

- 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:

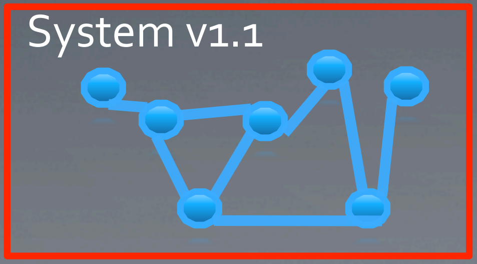


- A requirements specification is unambiguous if all its requirements are unambiguous.
- Example:
  - Build a system, any system... derp!

(Impossible to exemplify!)

Complete?

- A requirement does NOT have this attribute! A requirement cannot be complete! ONLY FOR SPECIFICATIONS!
- Think about what it means!



- 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?

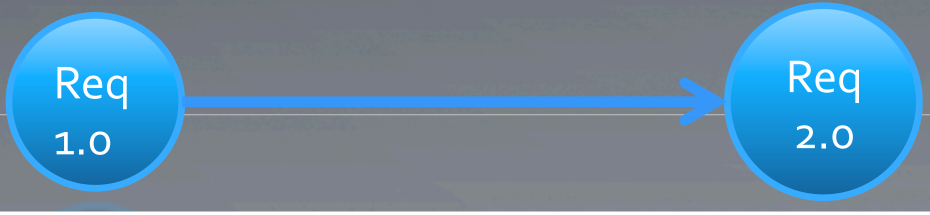
- A requirements specification is ranked if all requirements have an identifier to indicate either stability or importance.
- P – Priority, S – Stability
- Example:
  - P1, S∞, The system shall…
  - P∞, S1, The system shall…

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, i.e. The [actor] shall [action verb] [observable result]



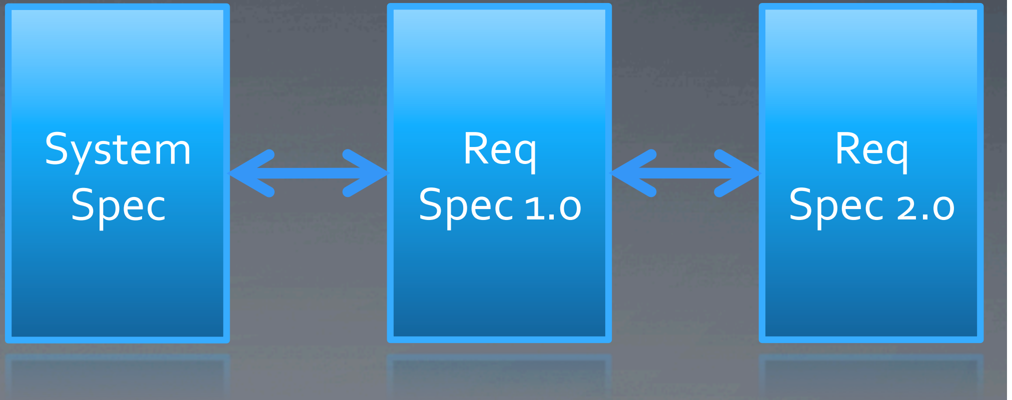
- An SRS is modifiable if its structure and style are such that changes can be done easily to the requirements whilst keeping the completeness and consistency of the specification, its structure and style.
  - Coherence
  - No Redundancy
  - Separate requirements

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.



But there are more!

- The ones above are defined in IEEE 830 for SRS’s.
- Davis defines a few more!
  - Challenges IEEE 830.
  - Mixes attributes for single requirements and for specifications! Be aware what’s what!

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.

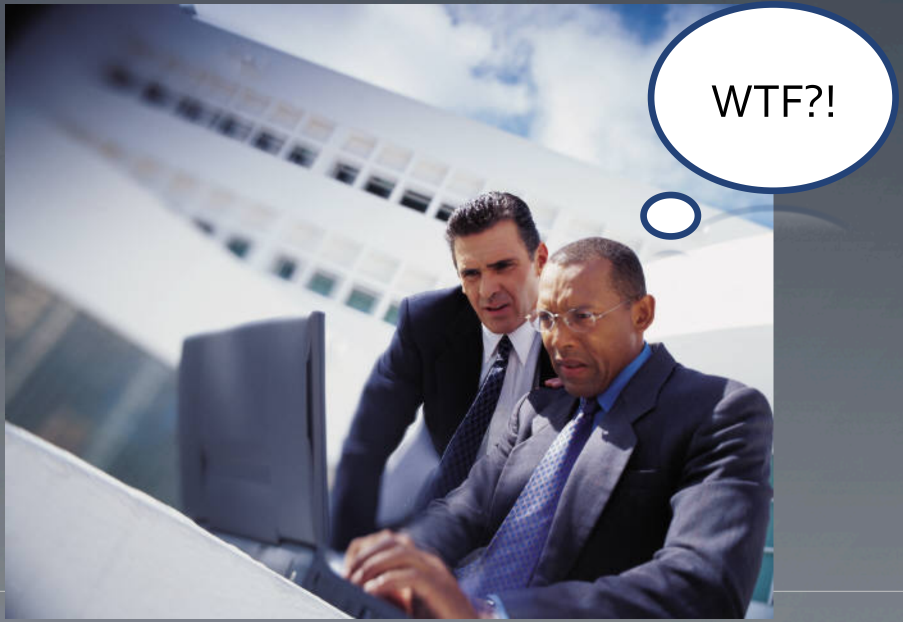
- A requirements specification is achievable if it is possible to construct a system including all the requirements from that specification.
- Example: One week before deadline, customer changes their mind that response-time of the system should be 4 ms instead of 40 and that ALL the UI components must be changed…
- What do you do?
  1. Work 25 hours a day?
  2. Tell the customer to go BEEEEP themselves?

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.

The IEEE 830 standard

- Defines how to write a requirements specification
- Includes statements such as: (Related to Ranking of Reqs.)
  - "Have customers give more careful consideration to each requirement, which often clarifies any hidden assumptions they may have."
  - Ambiguity!



WTF? :P

What should be documented in a SRS?

- Introduction
  - Purpose
  - Scope
  - Definitions, acronyms, and abbreviations
  - References
  - Overview
- Overall description
  - Product functions
  - User characteristics
  - Constraints
  - Assumptions and dependencies
- Specific requirements
- Appendixes
- Index

Introduction

- Purpose: Purpose of the SRS and the intended audience
- Scope:
  - The software products, by name
  - What the software products will and will not do
  - Application domain of the products
- Definitions, acronyms and abbreviations: List these
- Overview: Description of what the SRS contains
- References: List of references and sources

Overall description

- Defines the interfaces of the system, HW and SW
- Product functionality organized in an understandable way
- User characteristics of the intended user of the system
- Constraints that will limit the developers options
  - i.e. Regulatory policies, HW limitations, Interfaces, etc.
- Assumptions that may affect the requirements
  - I.e. that the intended operating system will in fact not support the system given chosen hardware.

Specific requirements

- The software requirements described on a level of detail that is sufficient to design the system as well as test that the implementation fulfills the requirements.
- Especially the requirements should:
  - Conform to the previously mentioned quality attributes
  - Contain references to earlier related documentation
  - Be uniquely identifiable
  - Be organized to maximize readability
- This section should also include external interfaces, functional and quality requirements, System modes, User classes, etc.

Appendixes

- Not always necessary but could include
  - Sample input/output formats, descriptions of cost analysis studies, etc.
  - Supporting or background information
  - A description of the problems the software solve
  - Packaging instructions for the code, etc.
  - And more.

Index

- Index and table of contents are important for quick navigation of the SRS.

Assignment 2

- Review another students Assignment 1 using a checklist of different quality attributes.
- Answer the questions:
  1. What was good/bad with the requirements in the mini-SRS, and why, according to the checklist?
  2. What was good/bad with the mini-SRS as a whole?
  3. What was good/bad in the mini-SRS according to your own opinion?
- Note that the checklist only has brief descriptions of the quality attributes! You WILL have to read more about their different meanings.

Submission

- Max 3 pages!
- Written as a technical paper either in Latex or following the Word template on the course homepage.
- Assignment is individual!
- Use your anonymous codes, no names!
- Submission done through the FIRE system
- Deadline 9/9 at 18.00