Software Requirements Specifications

Dr. Julie Greensmith

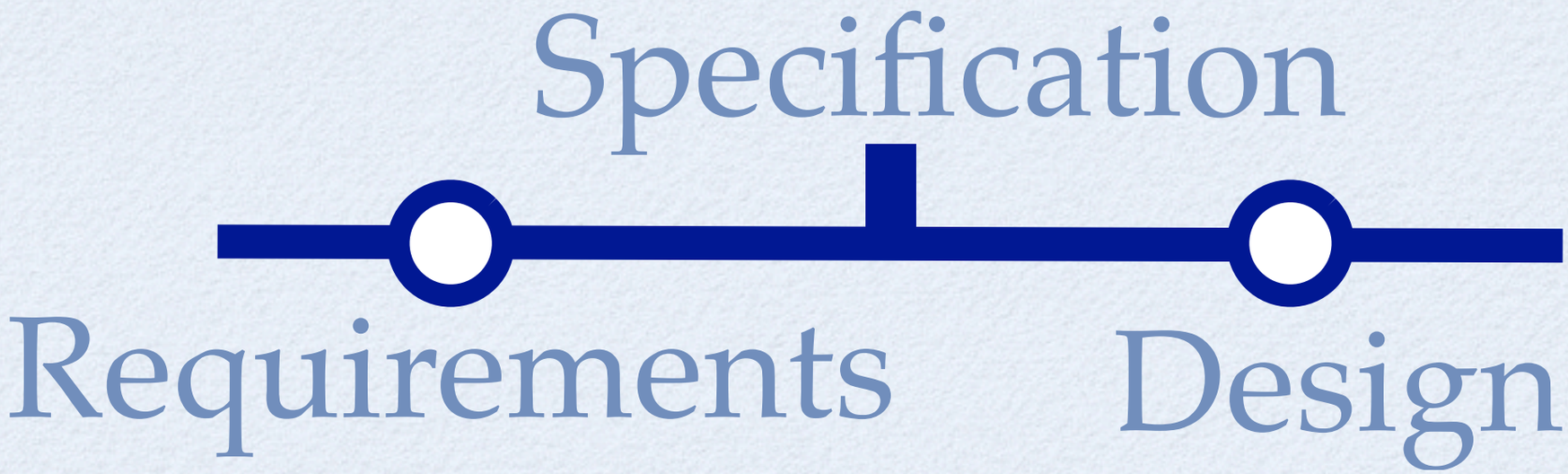
Coming up today...

- Generating requirements
  - revision of requirements gathering
  - feasibility for software
  - validating requirements
- Creation of software requirements specifications
  - Functional and non-functional requirements
- Introducing agile based User Stories

Last week...

- Covered terms and general definitions
  - what is software engineering
  - how do we create systems which cope with *emergence*
- Introduction to methodologies
  - Waterfall methods
  - Iterative methods & component based development

There are numerous stages



Stakeholders

- The "Who" of software engineering
- Can be a number of characters
  - a person
  - an organisation
  - an institution
  - an *entity*

Understanding Needs

- Requirements involves understanding the needs and objectives of the user, stakeholder and organisation

- Problem is this:
  - *clients don't always know what they want!*
- Use requirements capture process to get requirements and convert this into a *specification*

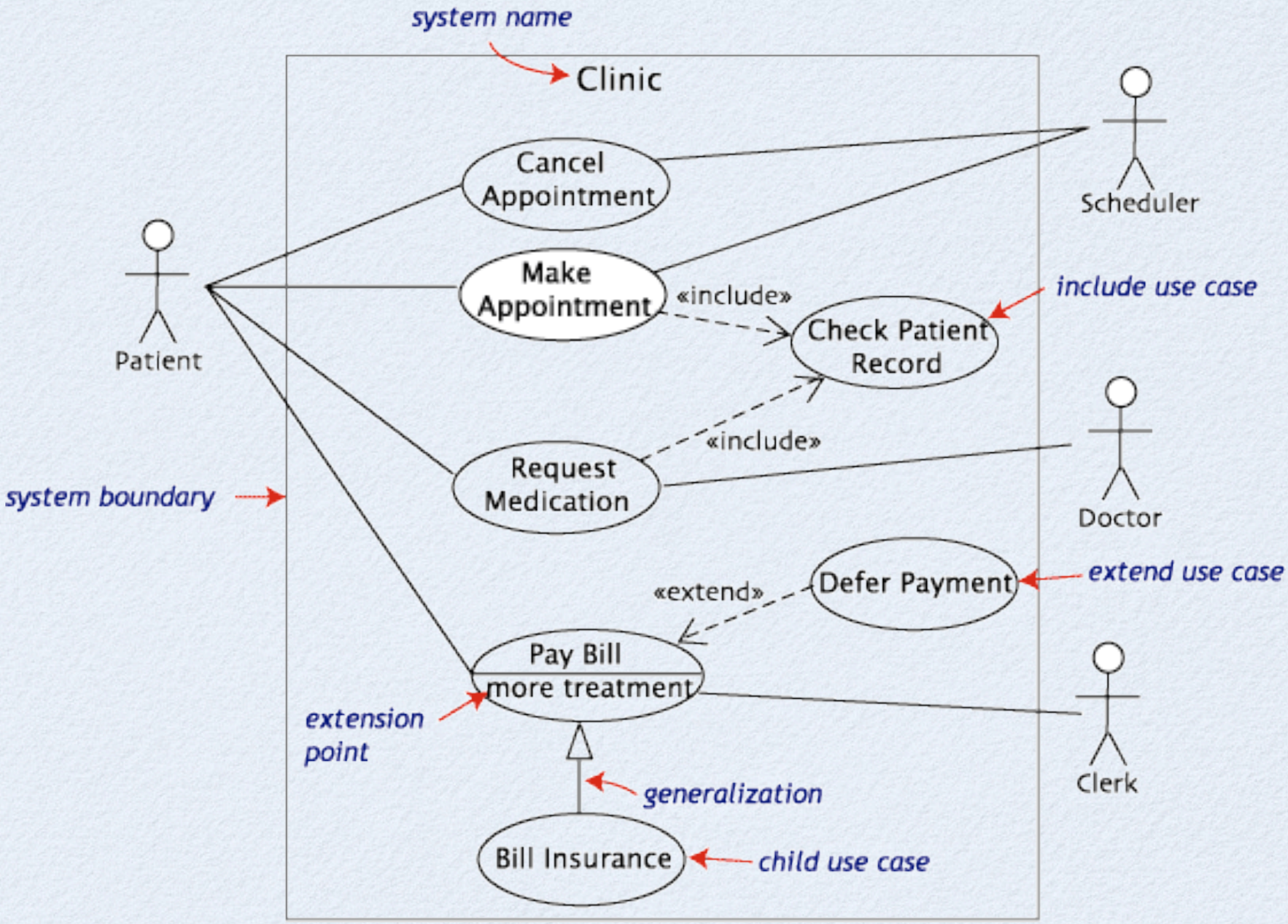
Stakeholders and Actors

- A stakeholder includes any party with vested interest in the project including:
  - Project leader, Project team members, Upper management, Project customer, Product user group, Project testers
- These people are stakeholders but not all of them are necessarily going to interact and use the actual system.
- An actor is someone who interacts and uses some part of the system
- This could be your project customer if it maybe enterprise software, or this could be a customer at a bank ATM

Use cases

- Describing the actions of the identified actors
- Expressed as a Use Case Diagram
- Linking actors with actions
- Each action drawn within a bubble

Use Case Diagram



Requirements Engineering

- The process of establishing the services that the customer requires from a system and the constraints under which it operates and is developed
- They are **functional** or **non-functional**:
  - Functional requirements describe system services or functions
  - Non-functional requirements is a constraint on the system or on the development process

#punsr **STAKEHOLDER**

Picking up meat

punsr.com

What is a requirement?

- Can be a number of representations
  - a high-level abstract statement of a service or of a system constraint to a detailed mathematical functional specification

- Requirements might serve a dual function
  - May be the basis for a bid for a contract - therefore must be open to interpretation
  - May be the basis for the contract itself - therefore must be defined in detail

Req and Spec go together

- Requirements **definition** is:
  - *"A statement in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers"*
- Requirements **specification** is:
  - *"A structured document setting out detailed descriptions of the system services. Written as a contract between client and contractor"*

Therefore....

- A software specification is:
  - *A detailed software description which can serve as a basis for a design or implementation. Written for developers to make the software*
- *Without understanding the requirements, the specification may be incomplete or inaccurate*

What do they look like?

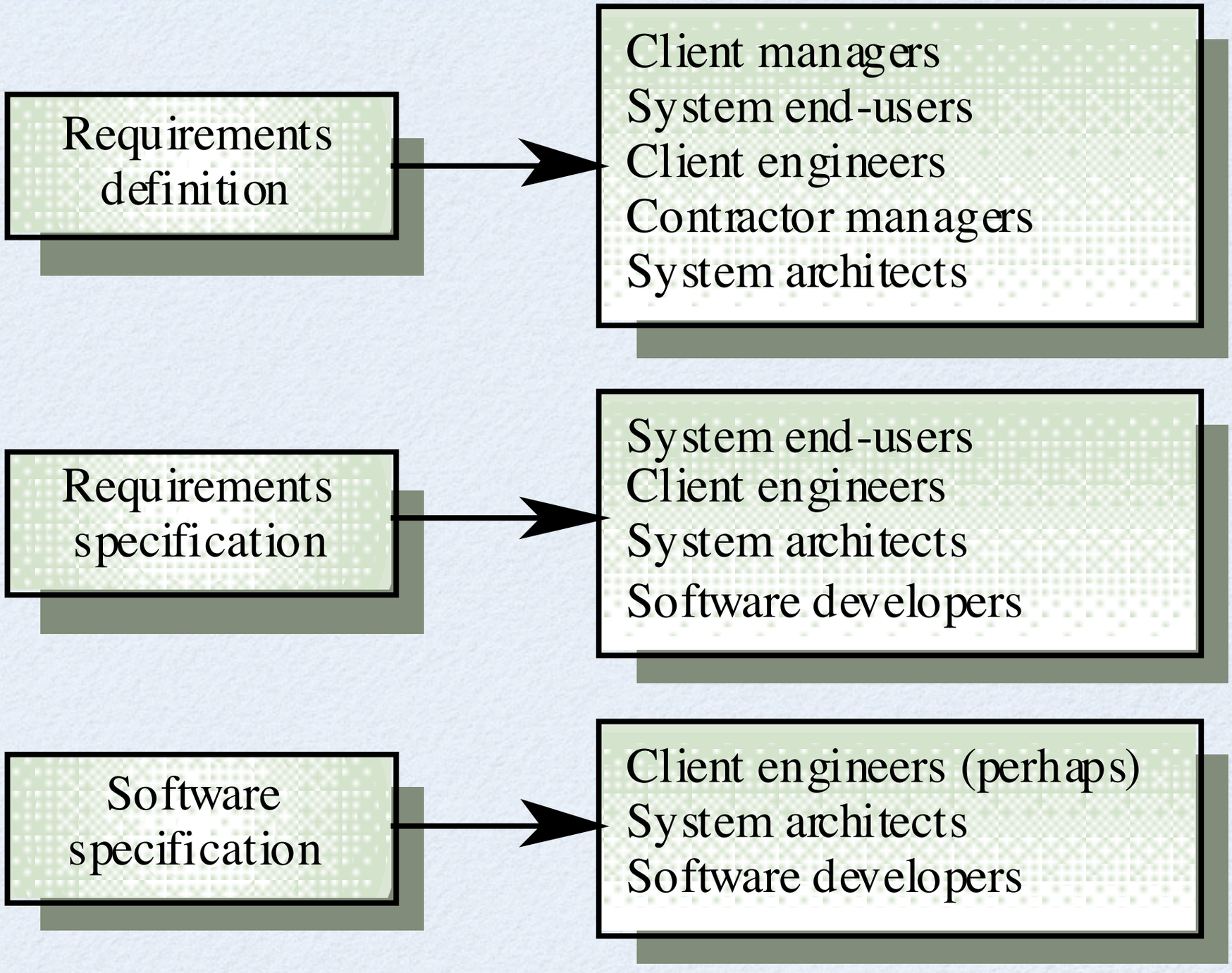
Requirements definition

The software must provide a means of representing and accessing external files created by other tools

Requirements specification

1. The user should be able to define the type of external file
2. Each external file type may have an associated tool which can be applied to the file
3. Each file is represented by a specific icon on the user display
4. When a icon is clicked, the tool is applied to the external file

Who cares?


Requirements definition
Client managers
System end-users
Client engineers
Contractor managers
System architects
Requirements specification
System end-users
Client engineers
System architects
Software developers
Software specification
Client engineers (perhaps)
System architects
Software developers

Tackle 'Wicked' Problems

Wicked Problem Characteristics

- There is no definitive formulation, can only anticipate the problems when you attempt to construct the solution
- They have no stopping rule
- Solutions are not true or false, they are good or bad
- There is no ultimate test of a solution
- Every solution is a one off solution as every problem is different, so no room to learn by trial and error

Cont....

- They do not have an exhaustively describable list of requirements
- Every problem is unique
- Can be considered the symptom of another problem
- How you chose to explain the problem limits the type of solution used
- The planner has no right to be wrong

Tackle 'Wicked' Problems

- Problems which are so complex that they can never be fully understood and where understanding evolves during the system development
- Therefore, requirements are normally both incomplete and inconsistent
- Iterative interplay between requirements and specification

Example Wicked Problems

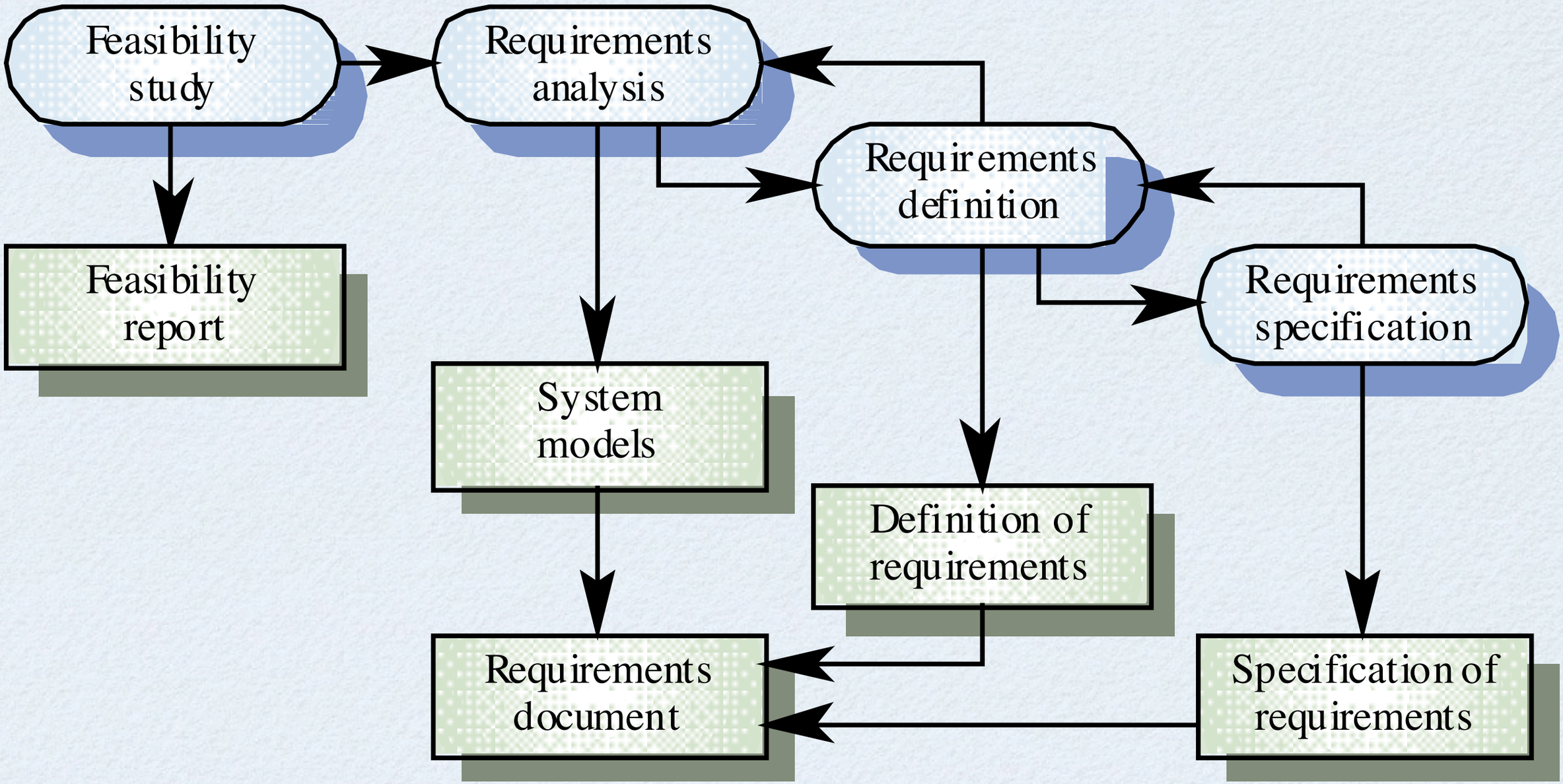
- Poverty, Climate Change, AIDS pandemic, global drugs trafficking given by the authors (Rittel & Weber)
- Applies to most software engineering projects
- NHS infrastructure
- TFL computer systems
- Aircraft Engine management systems
- Direct.gov systems

Inconsistencies happen

- Large software systems must improve the current situation
- It is hard to anticipate the effects that the new system will have on the organisation
- Different users have different requirements and priorities. There is a constantly shifting compromise in the requirements
- System end-users and organisations who pay for the system have different requirements
- Use prototyping approaches to *evolve*

Four stage process

- **Feasibility study**
  - Find out **if** the current user needs be satisfied given the available technology and budget?
- **Requirements analysis**
  - Find out what stakeholders require from the system
- **Requirements definition**
  - Define the requirements in a form understandable to the customer
- **Requirements specification**
  - Define the requirements in detail


Feasibility study
Requirements analysis
Requirements definition
Requirements specification
Feasibility report
System models
Definition of requirements
Requirements document
Specification of requirements

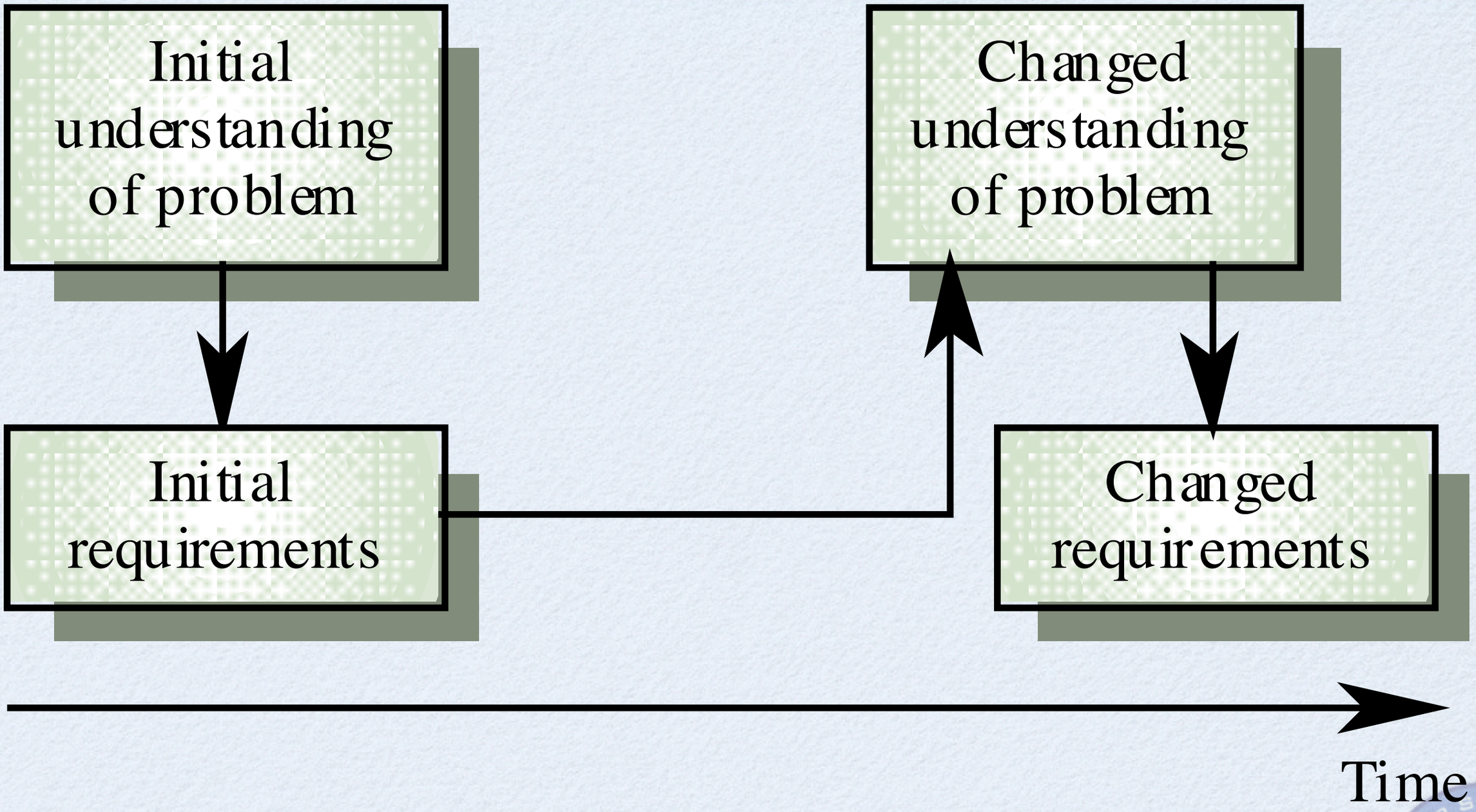
"The Requirements Document"

- The requirements document is the official statement of what is required of the system developers
- Should include both a definition and a specification of requirements
- It is NOT a design document. As far as possible, it should set of **what** the system should do rather than **how** it should do it

Validation of requirements

- Concerned with demonstrating that the requirements define the system that the customer really wants
- Requirements error costs are high so validation is very important
  - Fixing a requirements error after delivery may cost up to 100 times the cost of fixing an implementation error
- *Prototyping* is an important technique of requirements validation

Evolution


Initial understanding of problem
Initial requirements
Changed understanding of problem
Changed requirements
Time

Requirements analysis

- In order to create a document that your software engineers can use the requirements must be analysed by a requirements engineer
  - transformed into a set of specific statements
- Sometimes called requirements elicitation or requirements discovery
- Technical staff working with customers to elicit the application domain, services and operational constraints

Analysis has problems

- Stakeholders don't know what they really want
- Stakeholders express requirements in their own terms
- Different stakeholders may have conflicting requirements
- Organisational and political factors may influence the system requirements
- The requirements change during the analysis process.
- New stakeholders may emerge as the system evolves

Prototyping can solve this

- Produce a specification from the potentially incomplete requirements so that a prototype can be made
  - easier for stakeholders to decide if the requirements specified are actually what they want
- To do this you have to create something that an engineer can go and create
- Many modelling and prototype development frameworks to formalise this process

So you have your requirements

- Use the IEEE Standard 830-1998
- Convert the requirements into a specification
  - functional requirements
  - non-functional requirements
  - show how your systems should actually function

IEEE 830 - 1998

Table of Contents

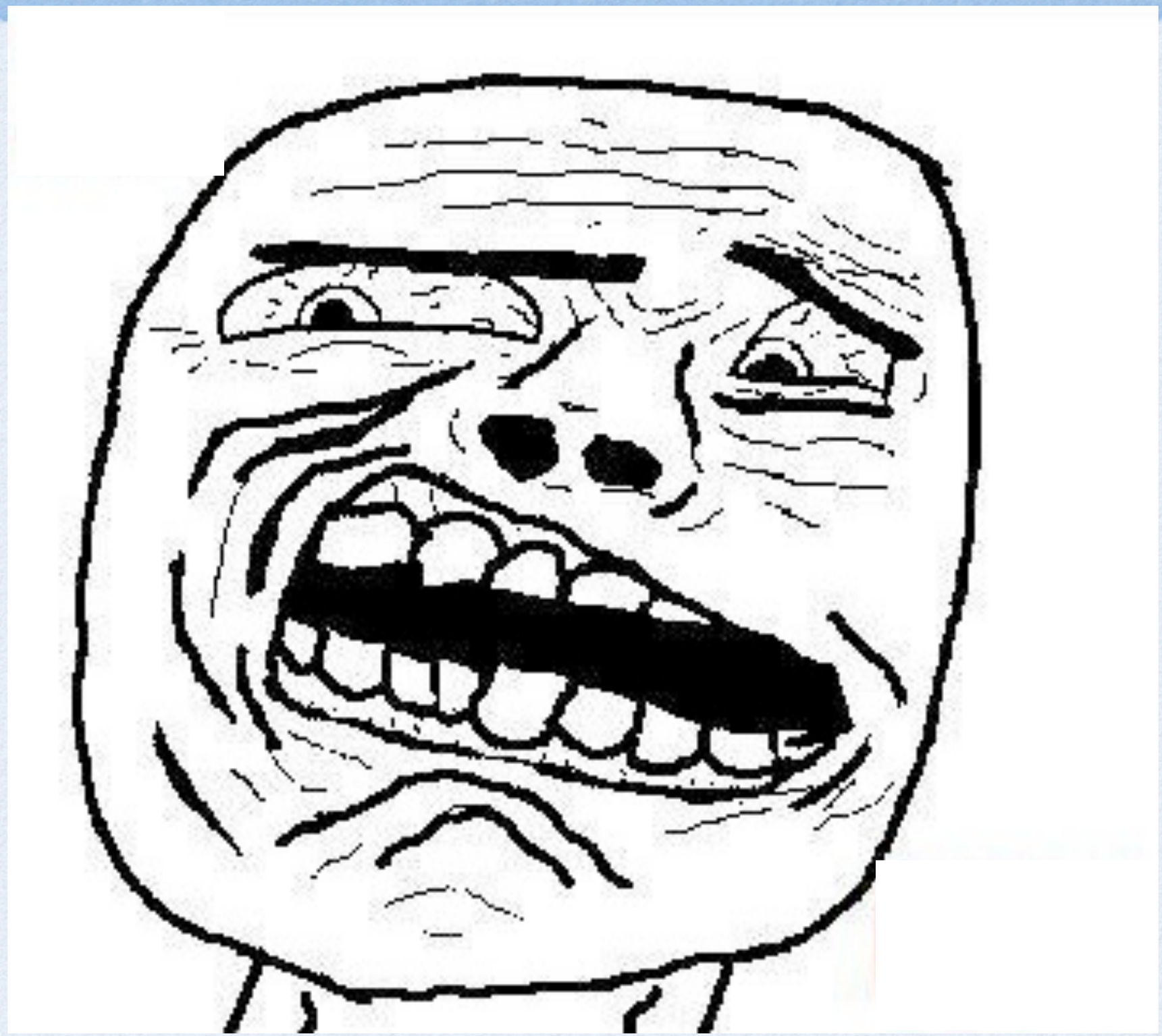
Producing an SRS Document

- SRS = Software Requirements Specification
  - Usually a big set of nested lists
- Contain a translation from anecdotal user wishes to a formal document
- Can evolve depending on your chosen SPM
- Also used as a checklist for testing purposes

To function or not to function

- Divide into functional and nonfunctional requirements
- **Functional:** "*A requirement specifies a function that a system or component must be able to perform*
  - *think:* objectives
- **Non-Functional:** "*A non-functional requirement is a statement of how a system must behave, it is a constraint upon the systems behaviour.*"
  - *think: constraints*

What does this mean???



Functional Requirements

- Technical specifications - languages, operations
- System design parameters and guidelines
- Data manipulation, data processing, and calculation modules
- Examples in a purchasing system:
  - supports the calculation of VAT exemption
  - calculates the estimated delivery time

Non-functional requirements

- Descriptive of the parameters of system constraints
  - quality attributes
  - desired reliability
  - security
  - predicted system cost,
  - constraints in design/implementation
- http://www.requirements.com/Glossary/NonFunctionalRequirements/tabid/91/Default.aspx

More Examples

- "The customer must place an order within two minutes of registering"
- "The customer must be able to access their account 24 hours a day, seven days a week."
- "Only the system administrator can view the credit card details"
- "System must monitor and display heart rate, blood pressure and EEC signal traces"

More Examples

- "The customer must place an order within two minutes of registering"
- "The customer must be able to access their account 24 hours a day, seven days a week."
- "Only the system administrator can view the credit card details"
- "System must monitor and display heart rate, blood pressure and EEC signal traces"

Agile Alternative

- Many agile methods argue that producing a requirements document is a waste of time as requirements change so quickly
- The document is therefore always out of date.
- Methods such as XP use incremental requirements engineering and express requirements as ‘user stories’
- Problematic for critical systems development

User stories for requirements

- "As a [role] I can [function] so that [rationale]."
- 1-2 sentence statements opening a dialogue with the user
- e.g. Traffic hotspots are highlighted on the map
- e.g. A customer can change his password
- e.g As a manager I want to edit worker timesheets

Understanding the syntax

- "Write game rules."
  - Better: "As a newbie game player, I want to know who goes first so we can start the game."
  - Better: "As a competitive gamer, I want a way to leapfrog my opposing players."
- "Play test the game.".
  - Better: Make testing, refactoring, etc. a default acceptance criteria on every Product Backlog Item

On the use of user stories

- Requires close collaboration with the customer and active user involvement in the development process
  - the customer has to be committed and have the time to engage in the process
- Requirements emerge and evolve flexibly
  - reducing predictability
  - harder to define a business contract and to secure a fixed price


THESE NO "TO DOS"
THESE USER STORIES!!!

The best way to get a feel for it...

- Go through scenarios and learn to extract:
  - Stakeholders
  - Use cases
  - Functional requirements
  - Non-functional requirements
  - User stories
  - *Will go through some practical examples on Wednesday*

Summary

- Generating requirements
  - revision of requirements gathering
  - feasibility for software
    - validating requirements
- Creation of software requirements specifications
  - Functional and non-functional requirements
- User Stories

Friday:

Generating Software Requirements Specifications

Read Sommerville Chapter 4