System Planning and Project Development (SYP)

Requirements

Unit 4

Every Project has Requirements

The hardest single part of building a software system is deciding precisely what to build

- From the customer's voice
  - Discussions
  - Observations
  - Unstructured data
- To a technical description
  - Detailed technical requirements
  - Interfaces to people
  - Interfaces to machines
  - Interfaces to other systems

Functional and Non-Functional Requirements

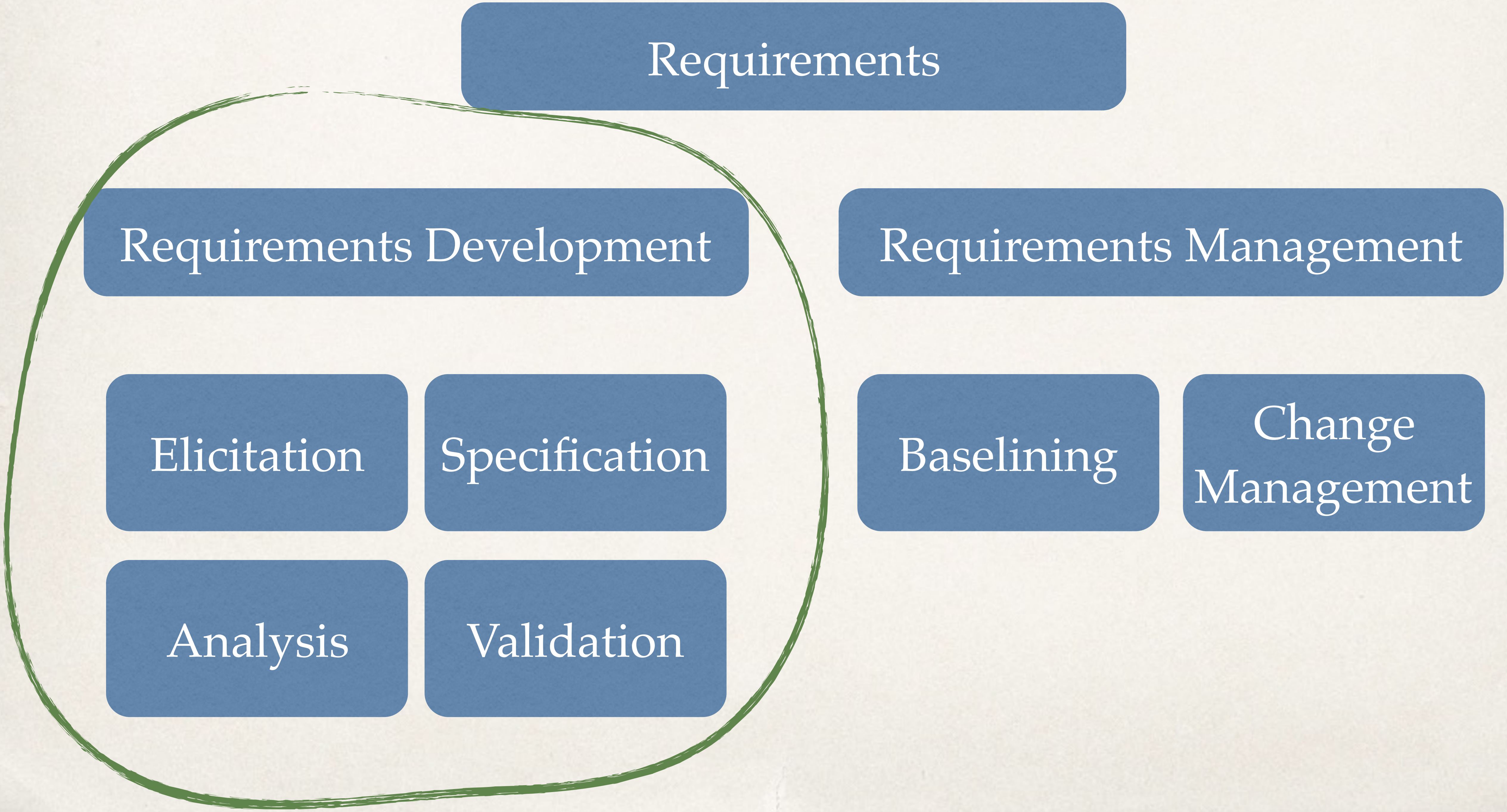
- Functional requirements
  - Features
  - Functions
- Non-functional requirements
  - Availability
  - Usability
  - Robustness

User’s Perspective

- Non-functional requirements
  - Maintainability
  - Portability
  - Reusability
  - Testability

Developer’s Perspective

Requirements Development and Requirements Management



Requirements

Requirements Development

Requirements Management

Elicitation

Specification

Analysis

Validation

Baselining

Change Management

Requirements Elicitation

Hearing the Customer's Voice

Interview

(Self-)Observation

Elicitation Workshop

Date of
Do you Smoke/use:
✓ Cigarettes (pks./day) 2
□ Used to smoke but quit in
✓ Alcohol (drinks/week) 4
□ Substance Abuse □ Yes ☒ No
Describe:
Questionnaire

Document study

Requirements Analysis

Requirements Development

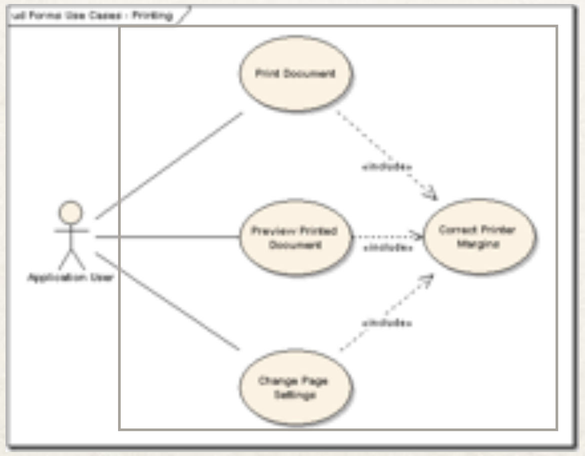
Elicitation | Specification

Analysis | Validation

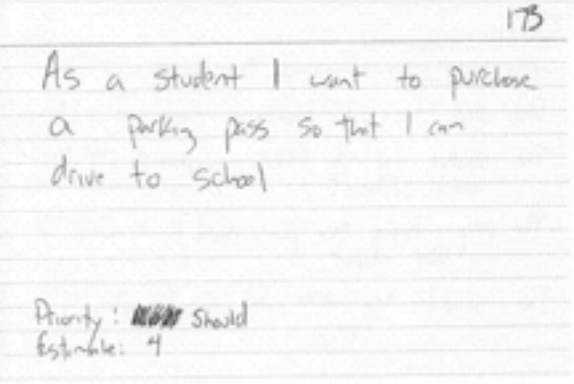
Domain Analysis

Prototyping

Speaking the Customer's Language



Use Cases



175

As a student I want to purchase a parking pass so that I can drive to school

Priority: Should
Estimate: 4

User Stories

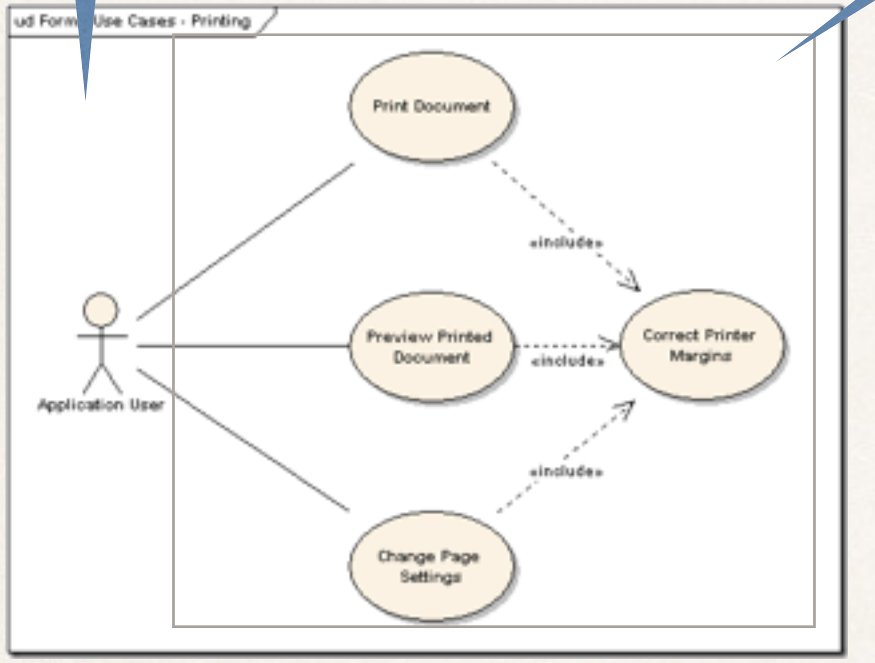
Use Cases / User Stories

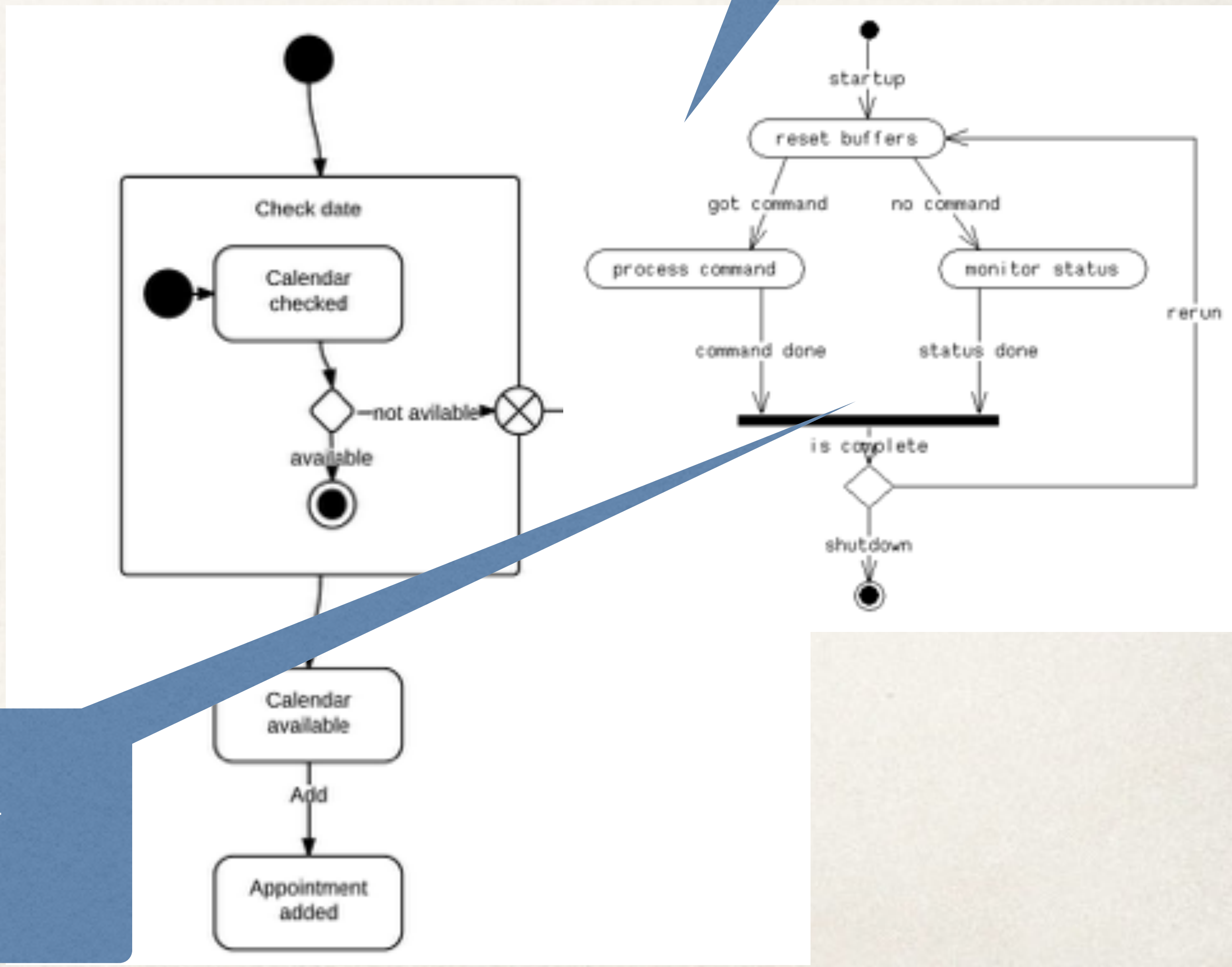
Analyze the features of the planned system

Start with customer’s perspective

Concentrate on WHAT not HOW

Refine …

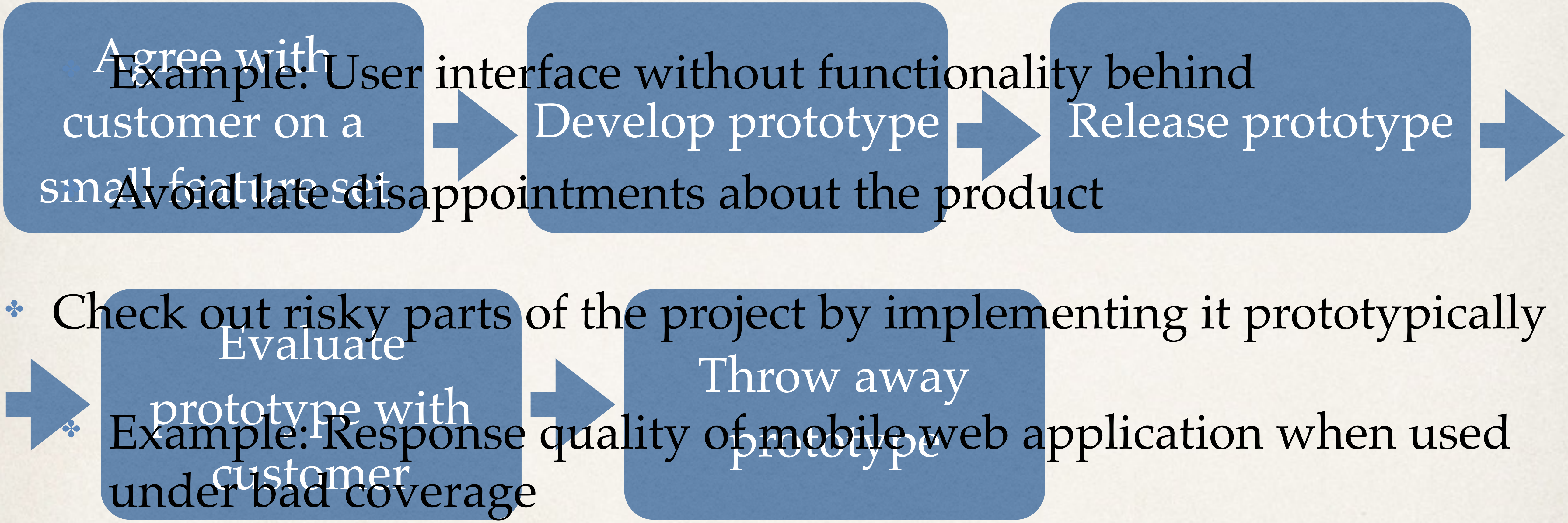




… to a technical perspective

Throw-Away Prototypes

- Make the user feel, how the final product will look like
  - Example: User interface without functionality behind
  - Avoid late disappointments about the product
- Check out risky parts of the project by implementing it prototypically
  - Example: Response quality of mobile web application when used under bad coverage
  - Reduce risk of project fail in a late part



Agree with customer on a small feature set

Develop prototype

Release prototype

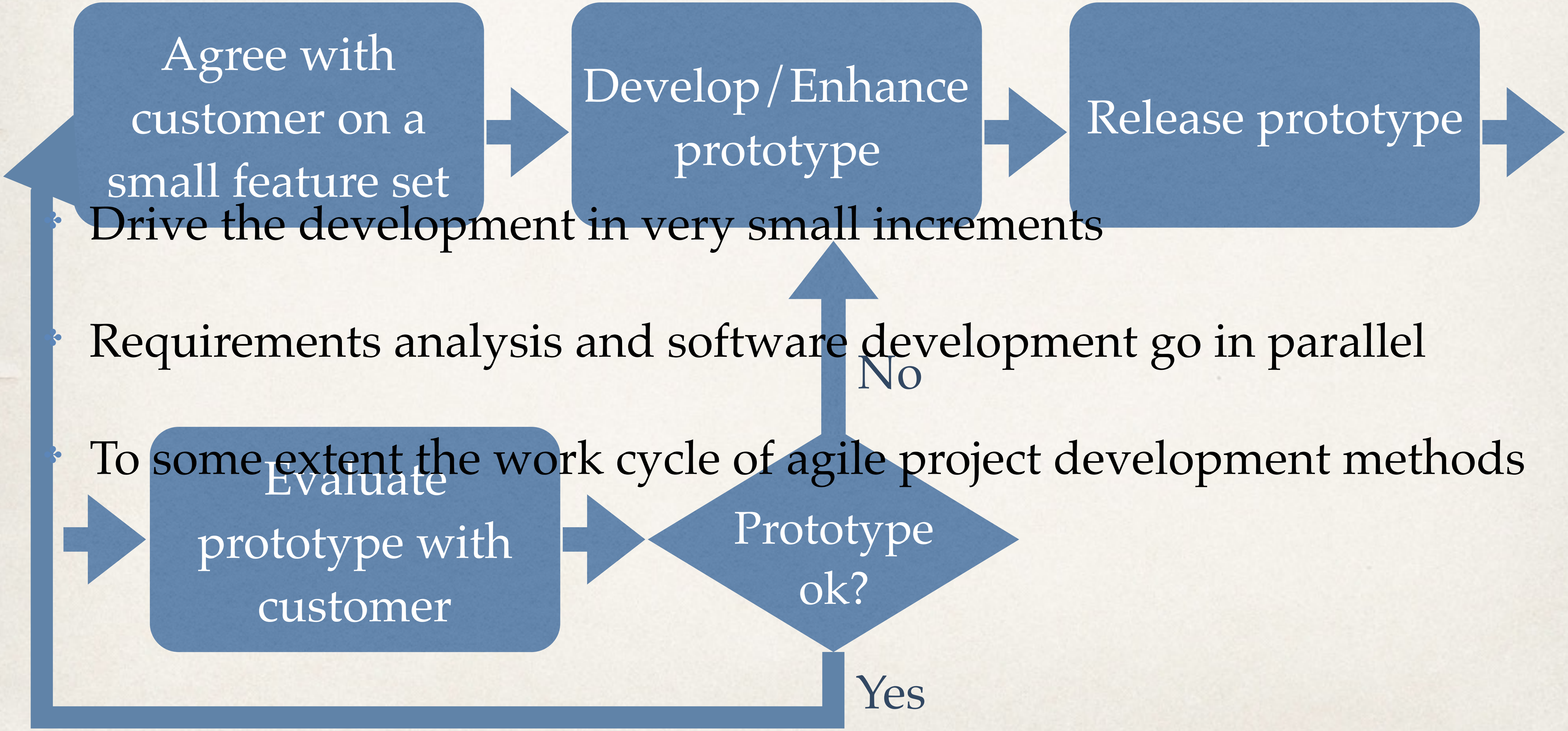
Evaluate prototype with customer

Throw away prototype

Requirements Development

Elicitation | Specification
Analysis | Validation

Evolutionary Prototype


Agree with customer on a small feature set
Develop/Enhance prototype
Release prototype
No
Evaluate prototype with customer
Prototype ok?
Yes

- Drive the development in very small increments
- Requirements analysis and software development go in parallel
- To some extent the work cycle of agile project development methods

Requirements Specification

- Document the requirements
    - Domain Analysis
    - Functional
    - Non-functional
    - Quantities
    - Embedding into and interfaces to existing infrastructure
    - Acceptance criteria

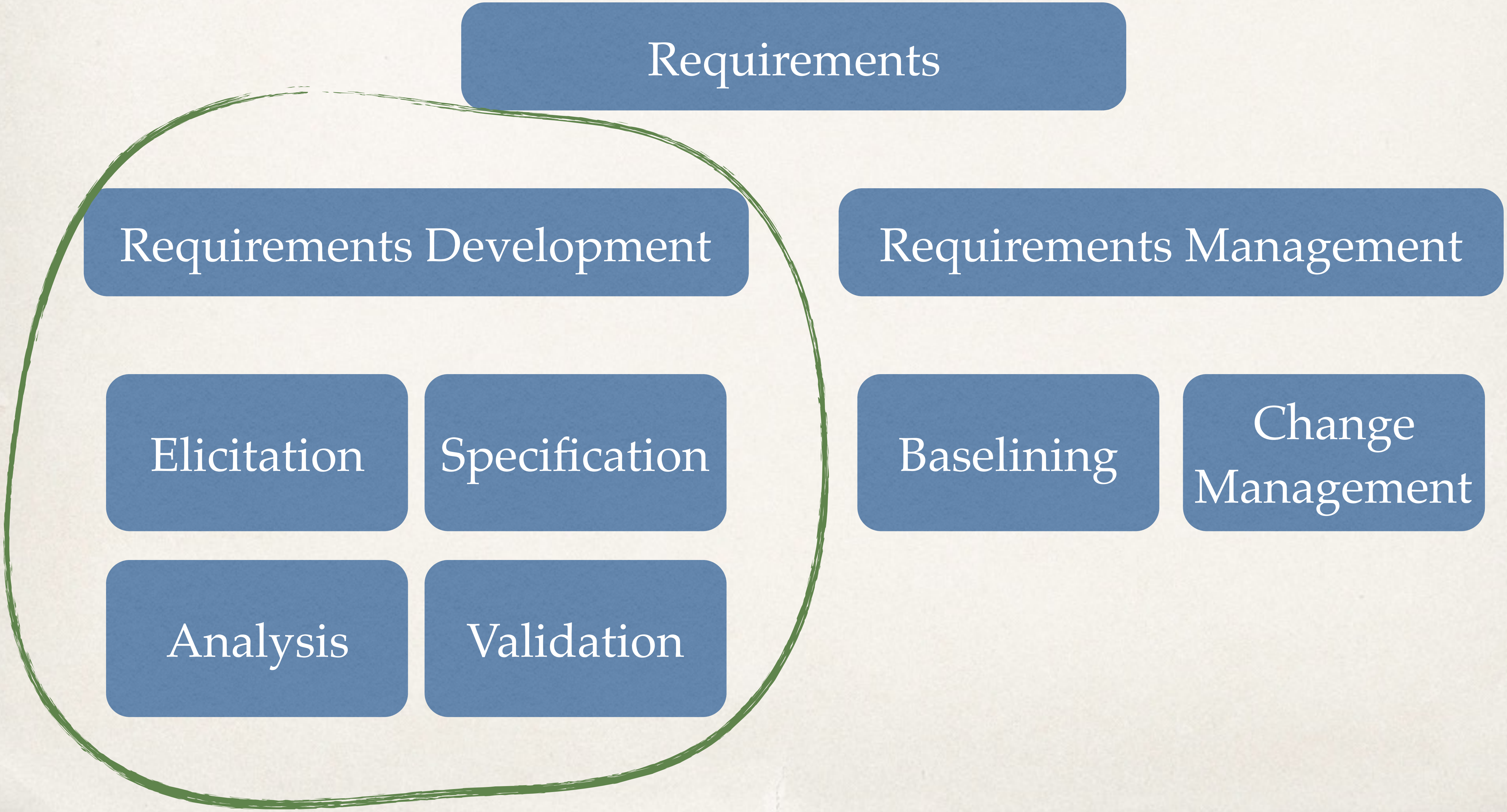
Validating Requirements

- Requirements Specification to be reviewed thoroughly WITH the customer
- Most effective: Write acceptance tests
    - Based on use cases / user stories
    - Normally business of the customer
    - Cuts a clear line whether feature is “done” when it comes to implementation

Requirements Sign-Off

- Do not think waterfall
- Requirements are not developed and then done
- Interpretation of “Sign-off”
  - Requirements development is mostly done
  - Rules of requirements management now apply

Requirements Development and Requirements Management



Requirements

Requirements Development

Requirements Management

Elicitation

Specification

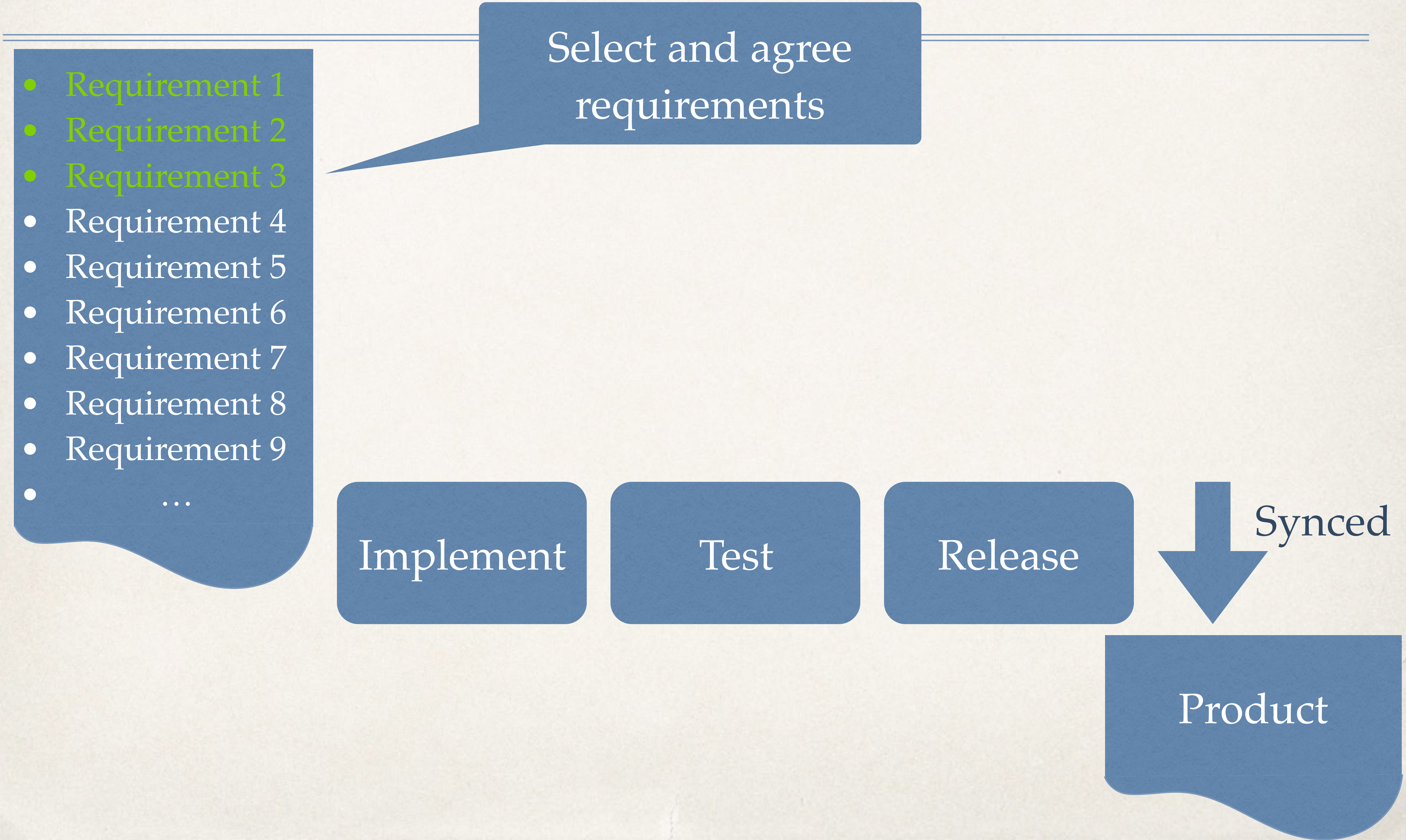
Analysis

Validation

Baselining

Change Management

Baselining – Increment 1



- Requirement 1
- Requirement 2
- Requirement 3
- Requirement 4
- Requirement 5
- Requirement 6
- Requirement 7
- Requirement 8
- Requirement 9
- …

Select and agree requirements

Implement

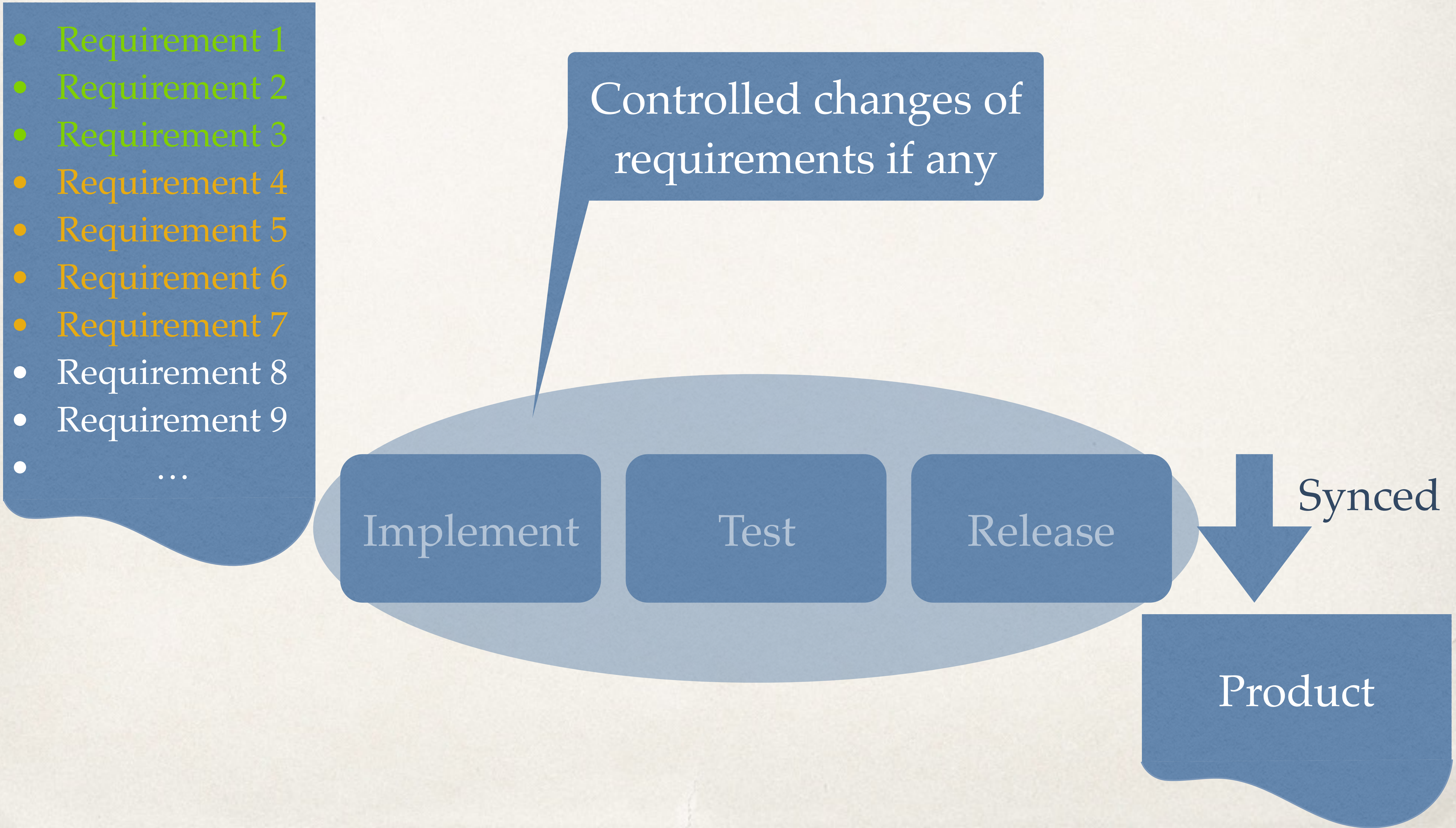
Test

Release

Synced

Product

Baselining – Increment 2



- Requirement 1
- Requirement 2
- Requirement 3
- Requirement 4
- Requirement 5
- Requirement 6
- Requirement 7
- Requirement 8
- Requirement 9
- …

Controlled changes of requirements if any

Implement

Test

Release

Synced

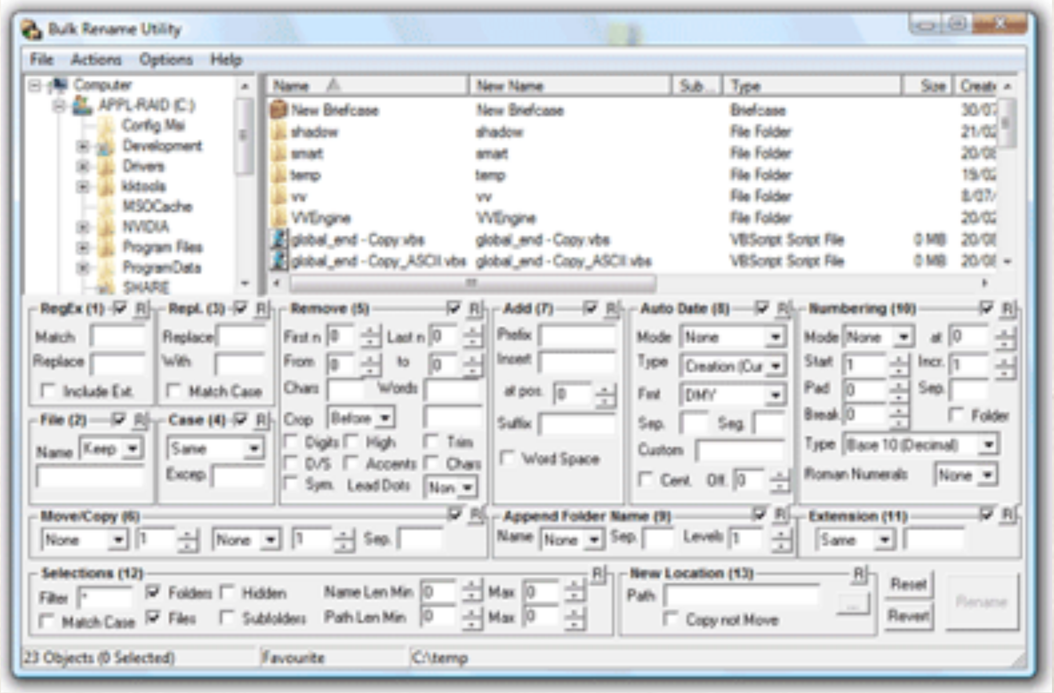
Product

Change Management

- The Times They Are A Changin’ (Bob Dylan, 1964)
- The Requirements They Are A Changin’ (Peter Bauer, 2012)

Healthy Change vs. Scope Creep

- (In software) requirements have to change
  - Best ideas come, when you have the first ideas in hand
- Nevertheless, creeping requirements at a late project stage may spoil a project seriously
  - Take care of a thorough requirements development
  - Learn how to say “NO”



Name	New Name	Sub..	Type	Size	Creat
New Briefcase	New Briefcase		Briefcase		30/07
shadow	shadow		File Folder		21/02
smart	smart		File Folder		20/0
temp	temp		File Folder		19/02
vv	vv		File Folder		8/07
VVEngine	VVEngine		File Folder		20/02
global_end - Copy.vbs	global_end - Copy.vbs		VBScript Script File	0 MB	20/0
global_end - Copy_ASCII.vbs	global_end - Copy_ASCII.vbs		VBScript Script File	0 MB	20/0

Change Control Board

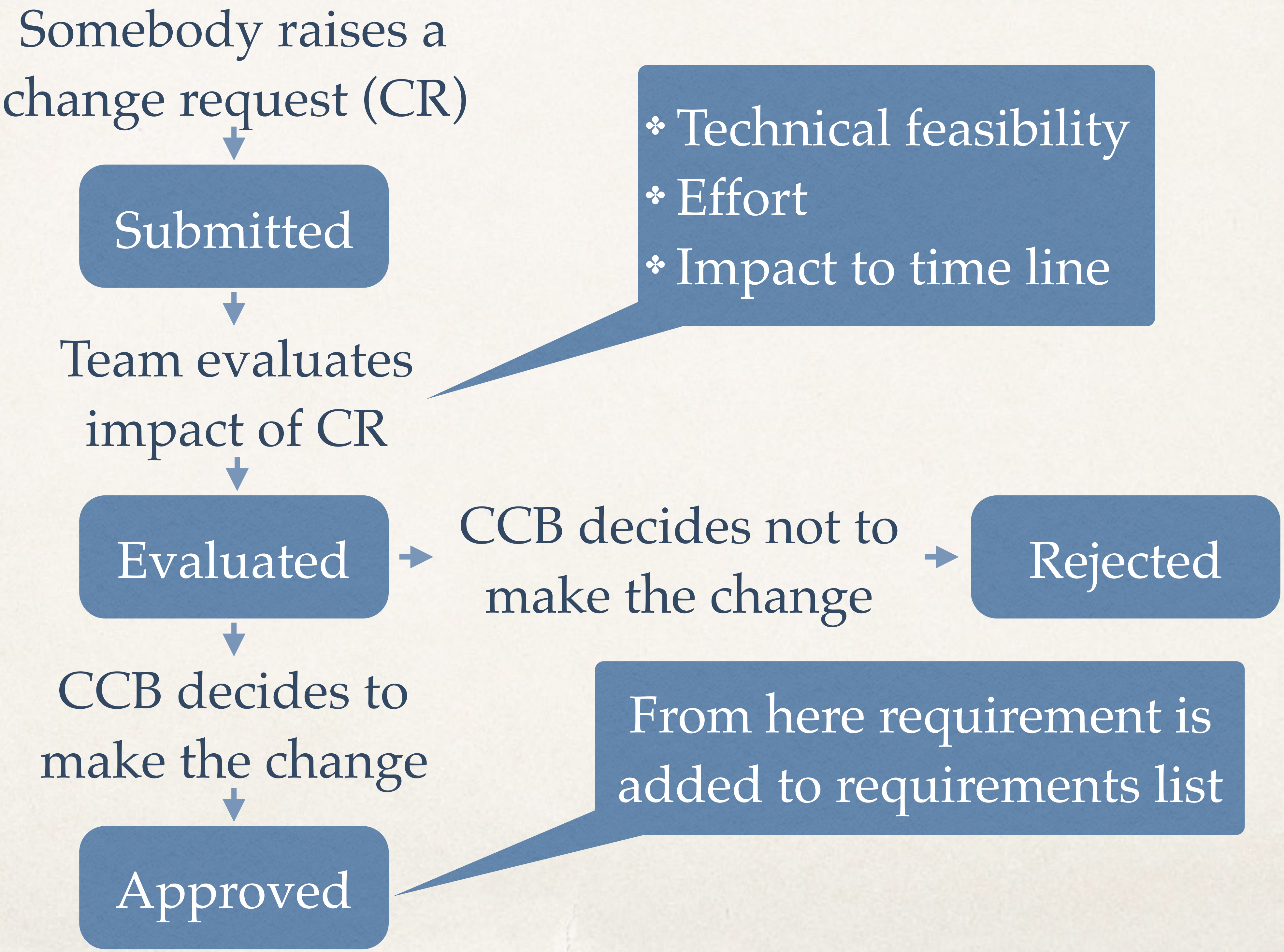
Requirements Management

Baselining

Change Management

- After requirements development is finished
- Requirements changes must be handled carefully
- Change Control Board (CCB) takes care of change requests (CR)
  - Customer
  - Product owner

Change Control Process



Somebody raises a change request (CR)

Submitted

Team evaluates impact of CR

- Technical feasibility
- Effort
- Impact to time line

Evaluated

CCB decides not to make the change

Rejected

CCB decides to make the change

Approved

From here requirement is added to requirements list

Summary

- Requirements are central to every software project
- Crucial is the translation from the customer’s language to a technical view
- Non-functional requirements must be considered as important as functional requirements
- A thorough requirements development is crucial for a successful project
- In later project stages a careful change management is necessary