

Requirements Management

Kari Systä

10.2.2014

Actual

- This weeks exercises are about Agilefant
 - The tools that will be used in the project
- All Weekly exercises 5 are at Lintula workstation room TC217.
Times are different from normal weekly exercise schedule.
YOU NEED AN ACOUNT TO LINTULA FOR USING THESE
WORKSTATIONS.**
- Tue 11.02.2014 at 10-12 and 12-14 o'clock.**
- Wed 12.02.2014 at 12-14 o'clock.**
- Thu 13.02.2014 at 12-14 and 14-16 o'clock.**
- Highly recommended if you do not want extra headache!
 - There has been questions about processing:
 - We assume that students have strong enough programming background to learn a new environment

Initial content of lectures

- Introduction
- Life-cycle models, their background
- Project management, product management, project planning – in general management aspects
- Scrum in details
- **Requirement elicitation, requirement management, requirements prioritization**
- Kanban, Customer Development and DevOps details
- Version management, configuration management, continuous integration
- Architecture issues, role of architect, architectural quality attributes, product families, (TIE-21300 will go deeper)
- Testing and quality assurance (TIE-21200 will go deeper)
- “Quality systems” and process improvement
- Embedded and real-time systems (other courses will go deeper)
- Safety-critical and dependable systems
- Effort estimation
- Software business, software start-ups
- Recap

Material I have used

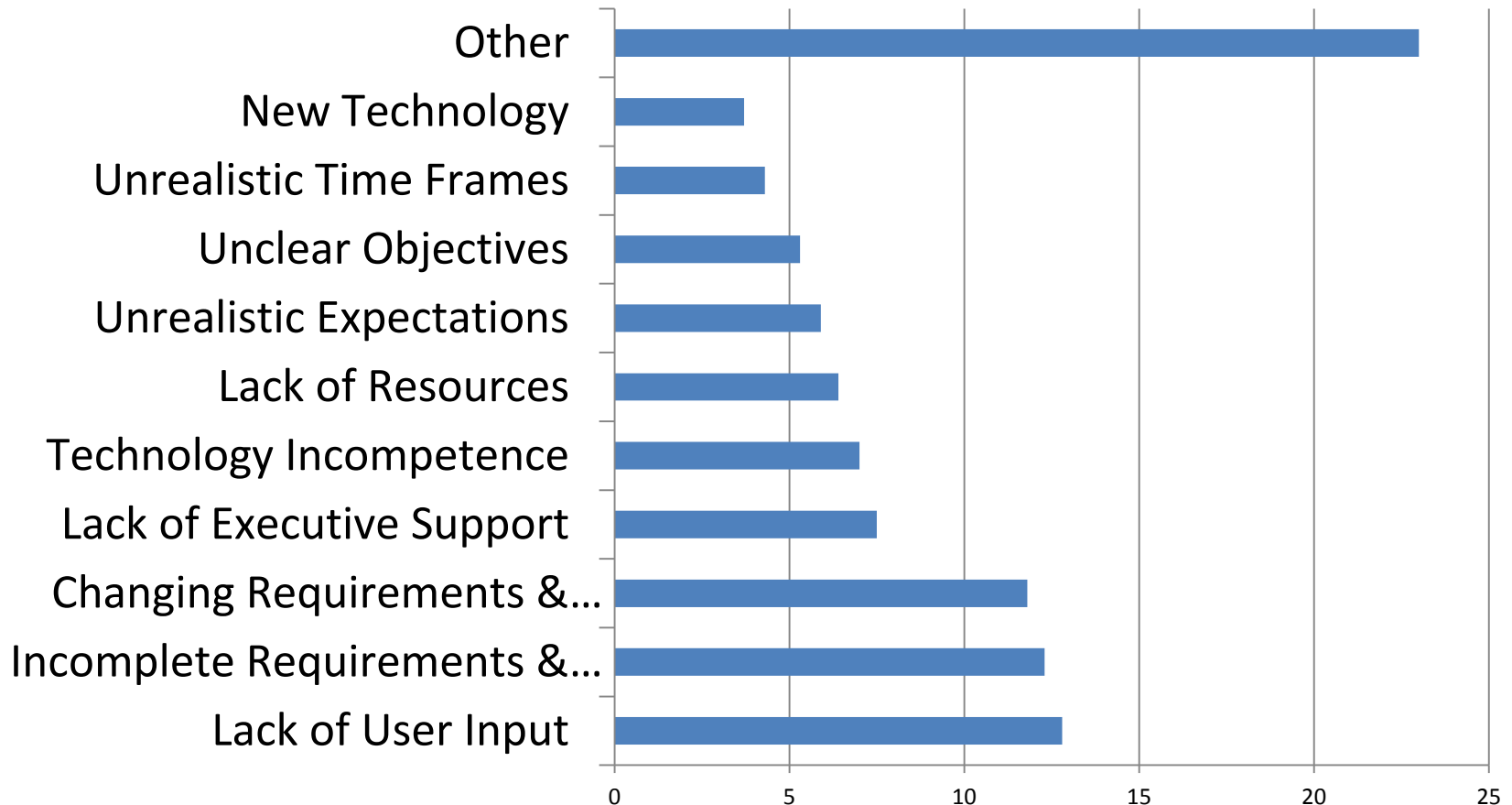
- Haikala & Mikkonen: ohjelmistotuotannon käytännöt
- TUT lecture slides
- Sommerville: Software Engineering, 9
- Davis: Just Enough Requirements Management (not required in exams beyond these slides)
- Miscellaneous Web resources and incidents I have seen

Motivation

(mistakes in requirements are expensive)

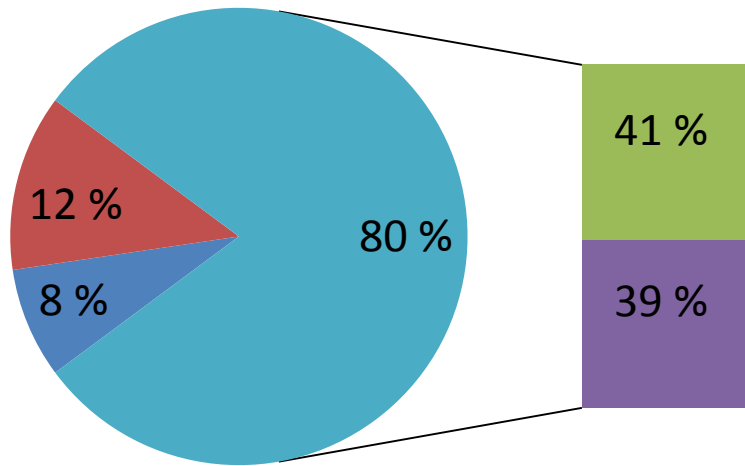
- Research shows that 40-50% of errors are related to requirements
- Requirements mistakes are very expensive

Reason of trouble n SW projects(%)



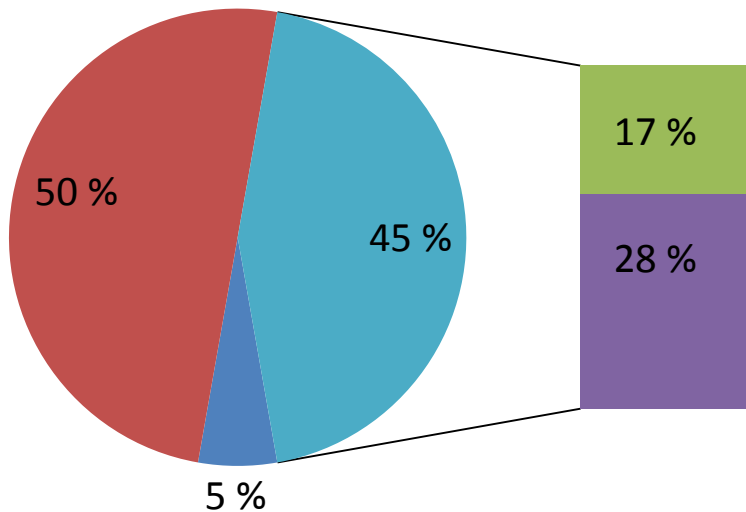
Source: The Standish Group. The Standish Group Report – CHAOS [WWW]. 1995, <https://cs.nmt.edu/~cs328/reading/Standish.pdf>.

Who is responsible for requirements



- Customers orders outside
- Under vendor steering
- Customer does
- Under customer steering

Customer view

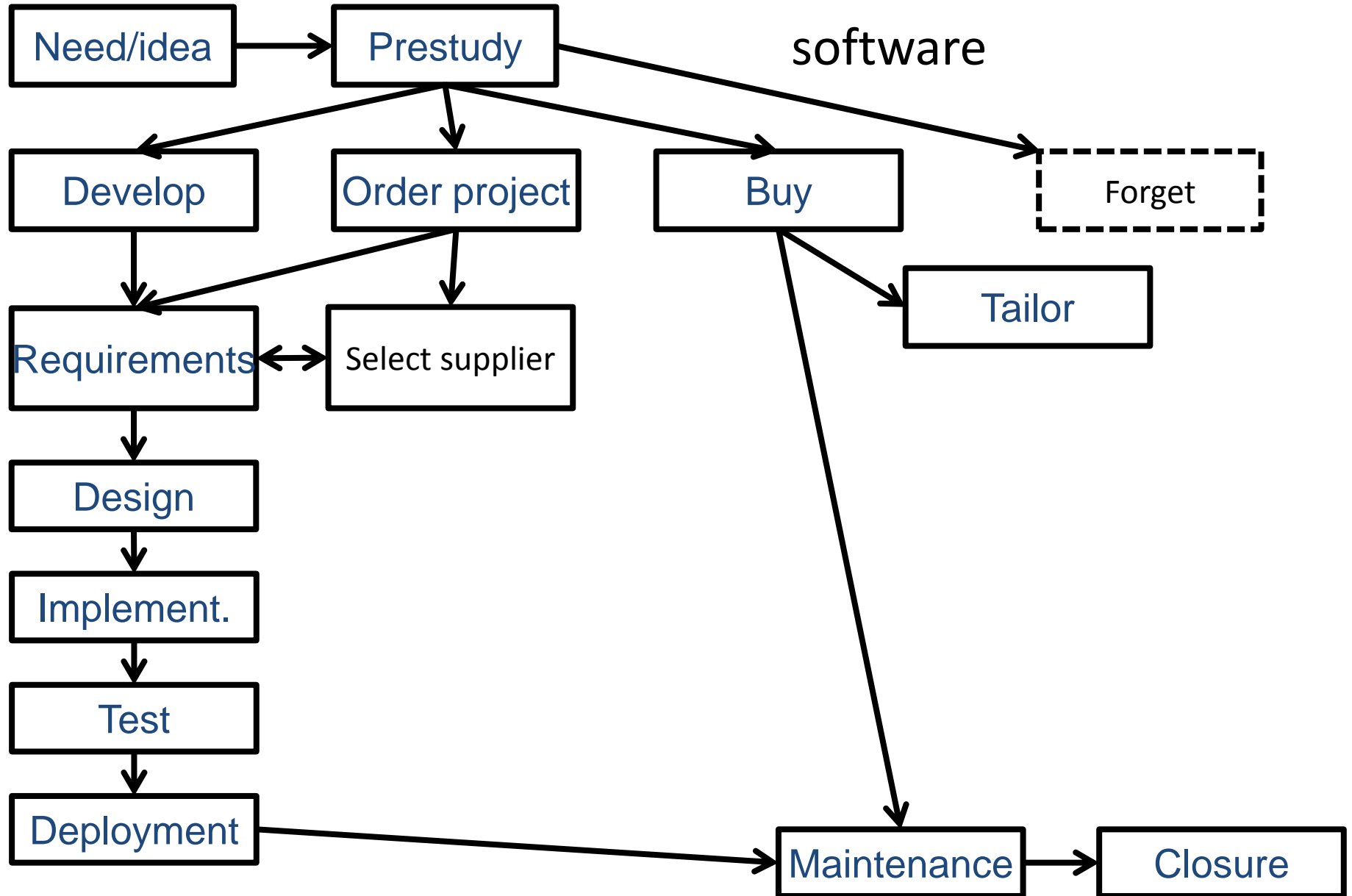


- Customers orders outside
- Under vendor steering
- Customer does
- Under customer steering

Vendor view

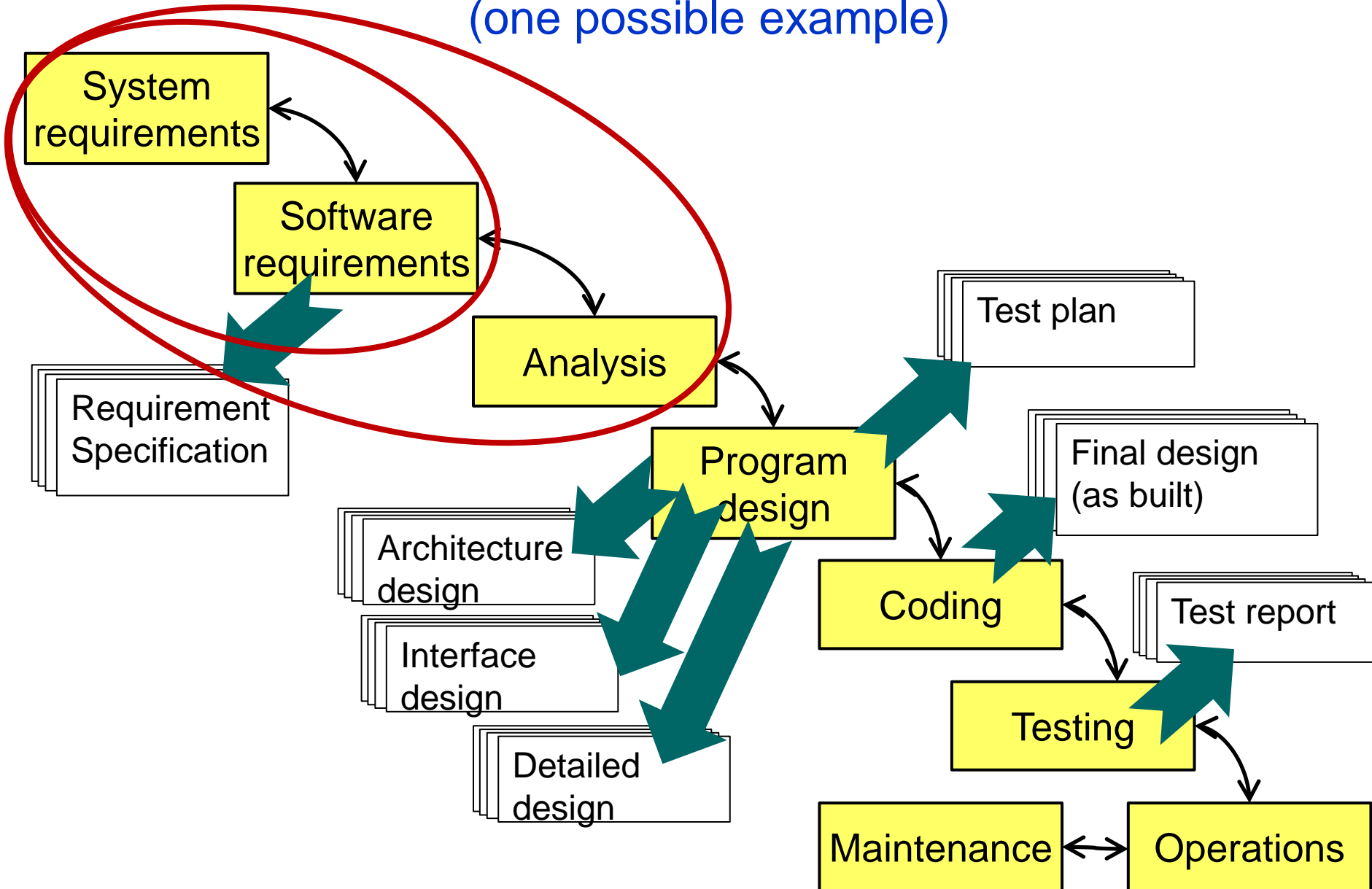
Source: master thesis of Erkka Vastamaa
TIE-22100/22106

From needs to
software



Documentation is a crucial part of waterfall

(one possible example)



Problems with waterfall

- Does not support division of the software to distinct stages
 - It is difficult to take out and use partial functionality
- Difficult to respond to changing customer requirements
- Management and motivation challenges of developers
 - Does not utilize full talent and motivation of talented and highly trained software developers
 - Does not show trust and empowerment
- Usually, waterfall is considers suitable for projects where
 - Requirements can be know in advance
 - Milestone reviews and audits are needed for example by security standards,

And will that work?

- Assumption 1: good requirements can be written if enough effort is put on them
 - But: customer needs change over the time – even during the project
 - But: software is abstract until it is seen and tried
- Assumption 2: changes are small
 - But: they are not (and address surprising parts)
- Assumption 3: Integration is as easy as glue components together
 - But: the components are implemented by humans
- Assumption 4: schedule is followed
 - Actually very seldom
- But it precise requirements can be agreed in advance waterfall might be the most efficient method.



How the customer explained it



How the Project Leader understood it



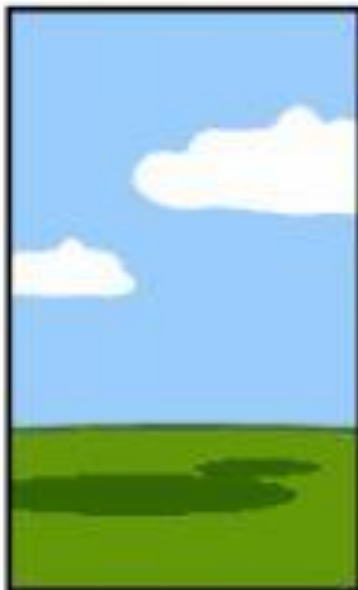
How the Analyst designed it



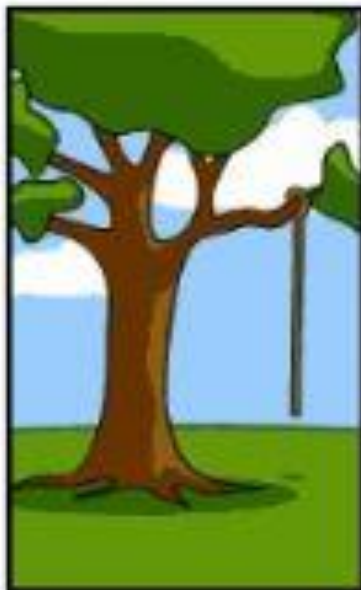
How the Programmer wrote it



How the Business Consultant described it



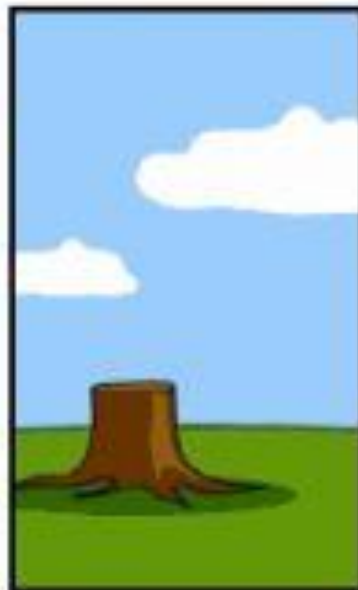
How the project was documented



What operations installed



How the customer was billed



How it was supported

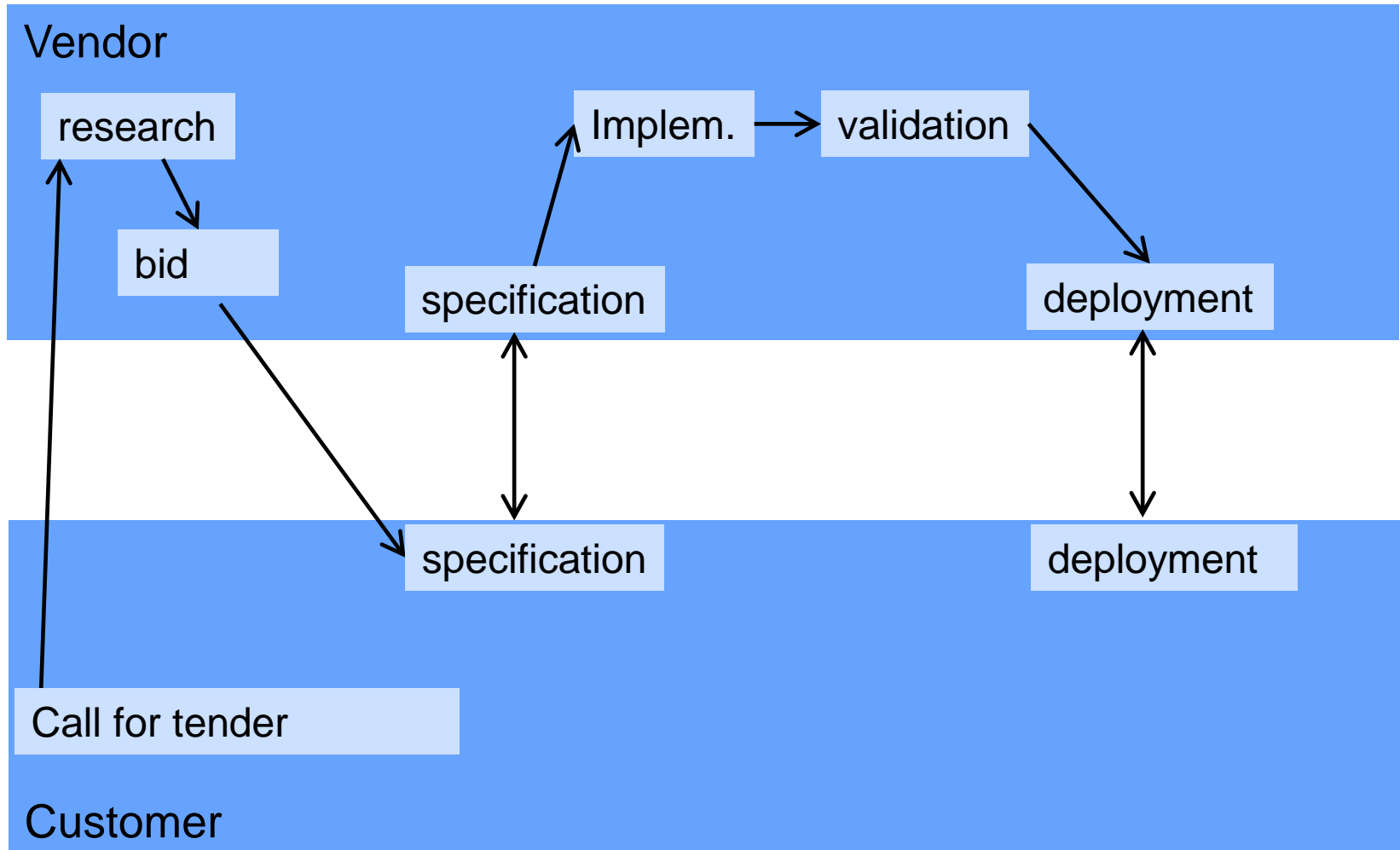


What the customer really needed

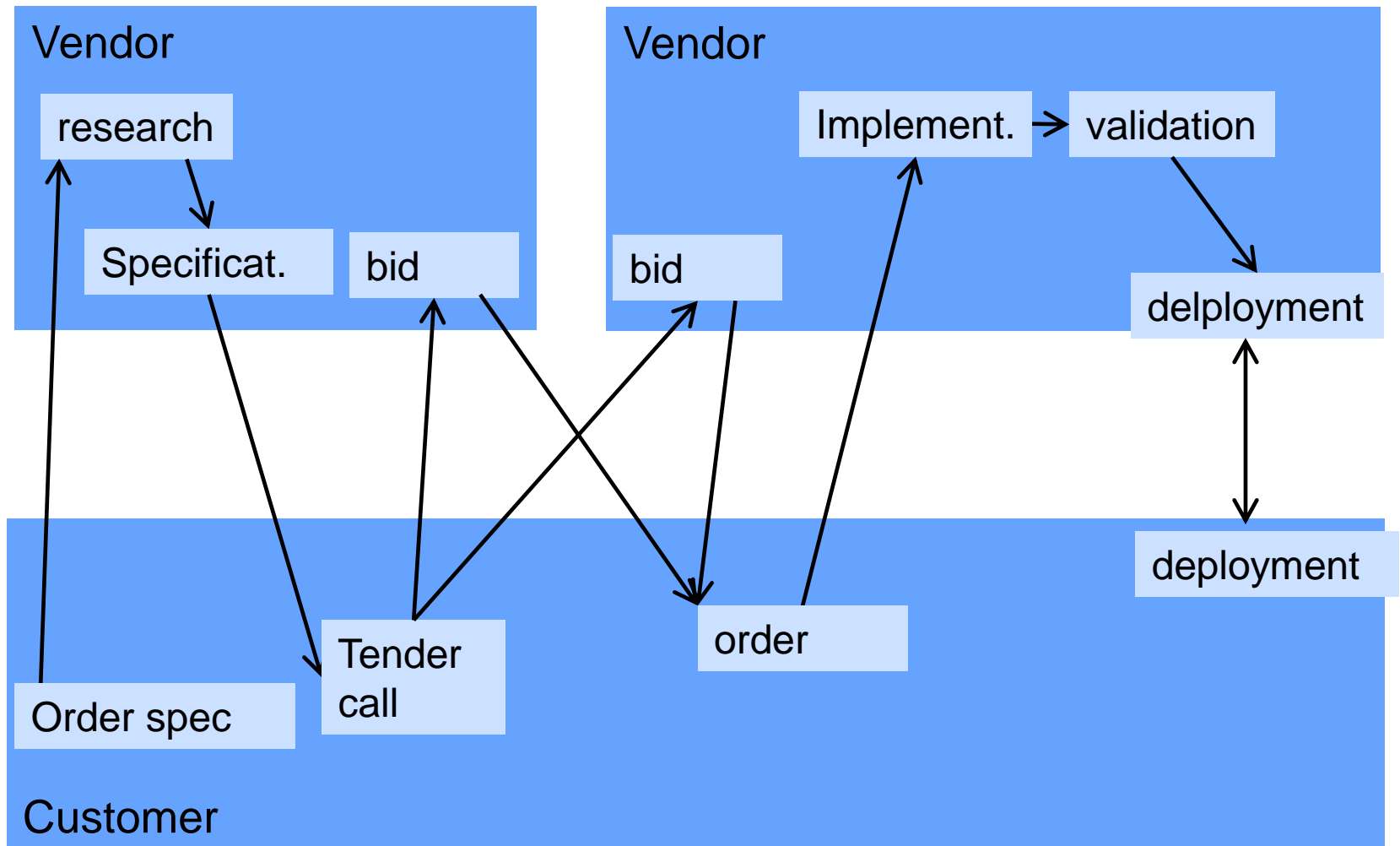
Will that work?

- Assumption 1: good requirements can be written if enough effort is put on them
 - But: customer needs change over the time – even during the project
 - But: software is abstract until it is seen and tried
- Assumption 2: changes are small
 - But: they are not (and address surprising parts)
- Assumption 3: Integration is as easy as glue components together
 - But: the components are implemented by humans
- Assumption 4: schedule is followed
 - Actually very seldom

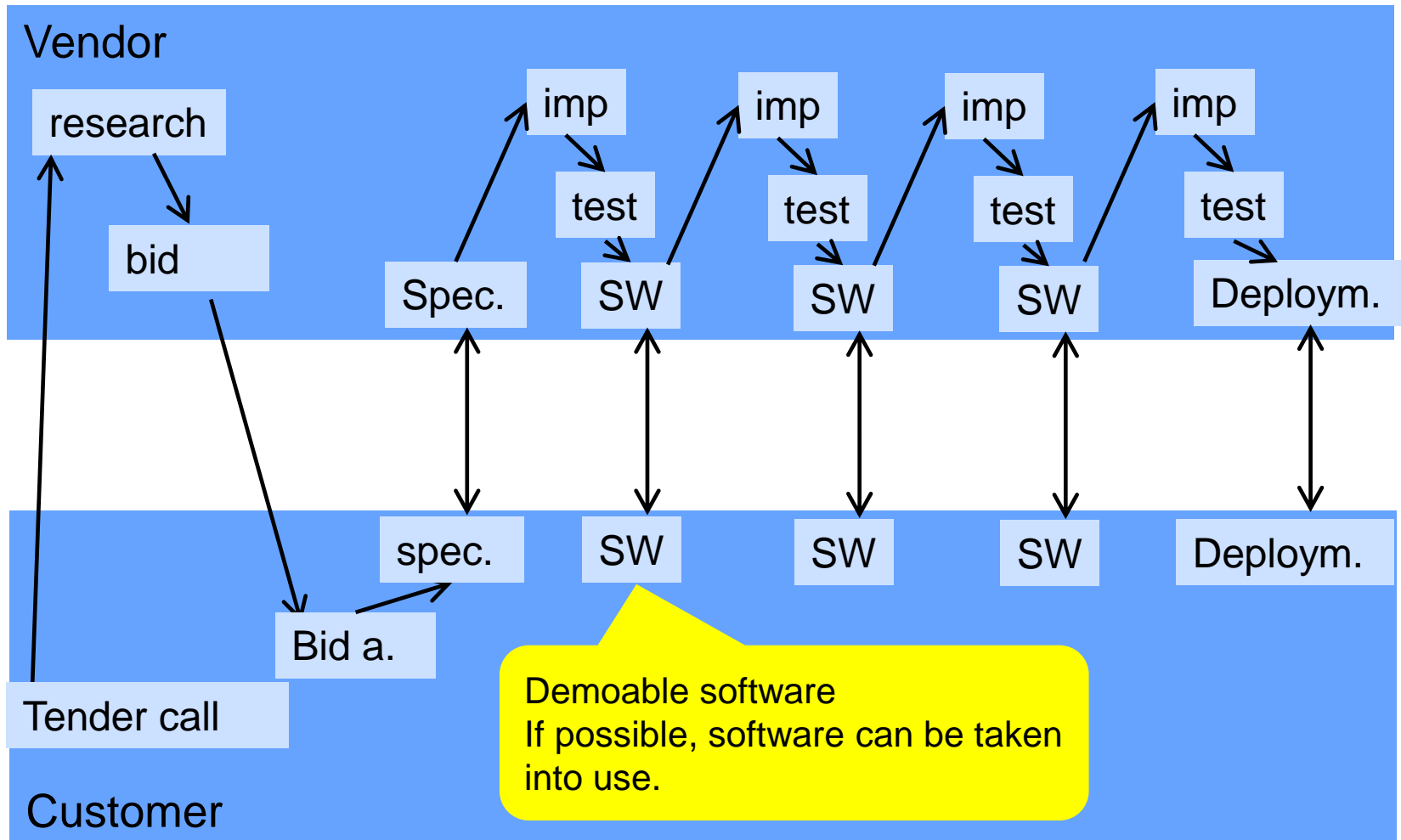
Different project – customer specific



Different project – customer specific



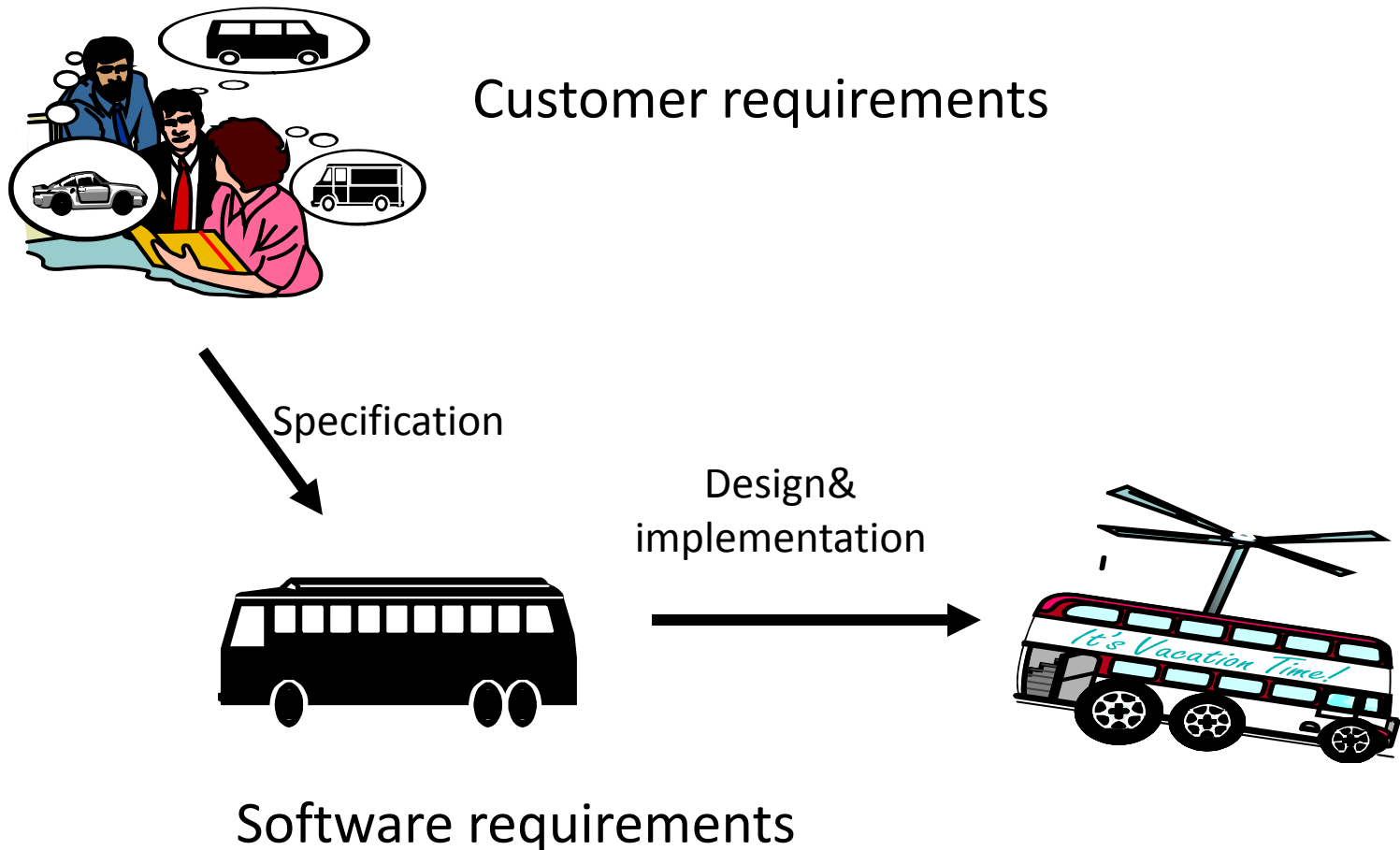
Iterative, agile



Two conflicting drivers

1. Spend enough time in finding, analyzing and documenting the requirement and everything will be easier later.
2. Requirements – or at least our understanding of them – will change anyways. So, lets plan the project to be as flexible as possible

From requirements to product



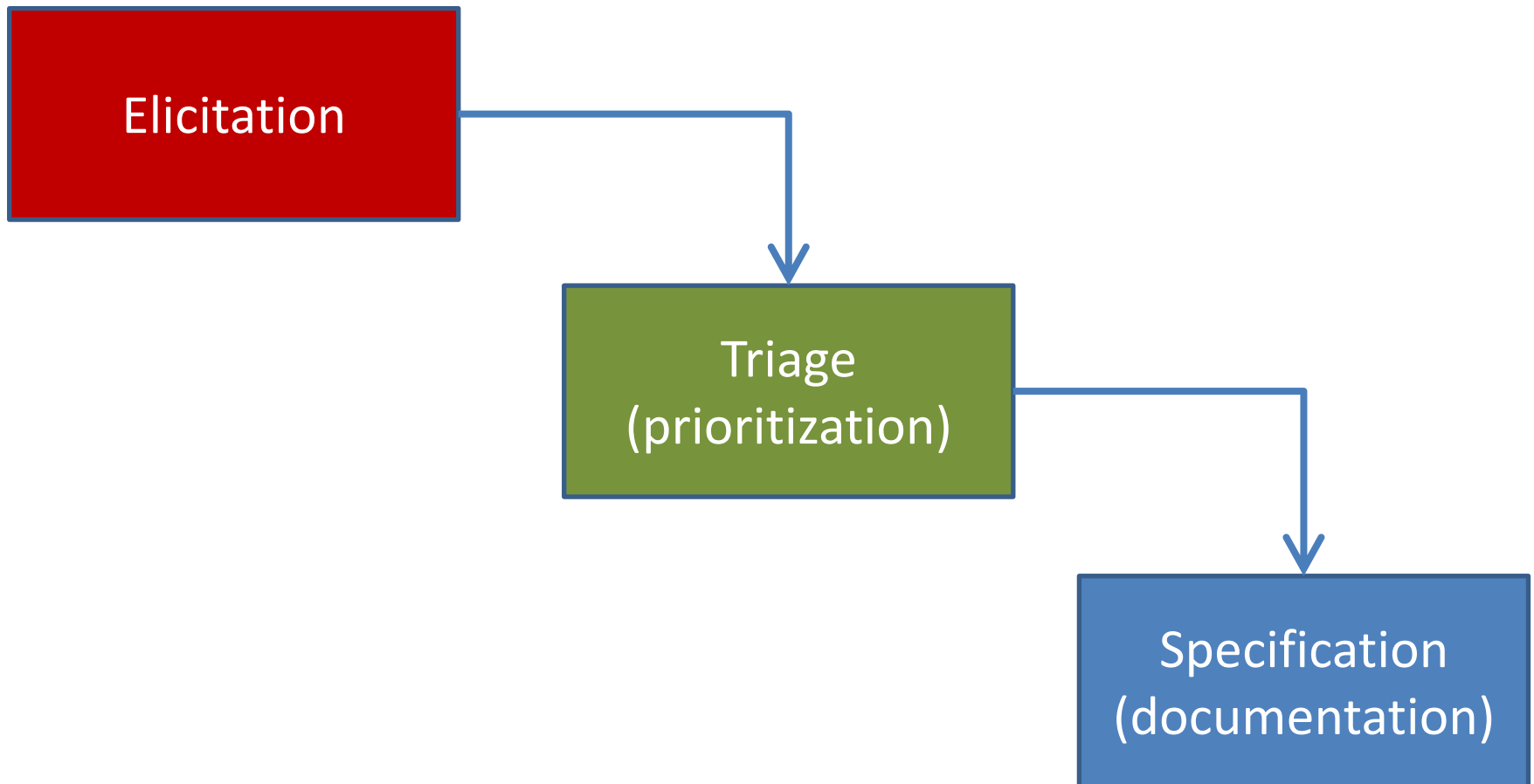
Different requirements - example

- Customer requirement – customer need or problem requiring solution: need to create error-free documents.
- Feature – distinguishable functionality from customer point of view: support for spell checking.
- Function – single operation of software: check spelling, propose corrections, correct automatically
- Technical requirements – implementation requirements: file buffer, user dialogs...

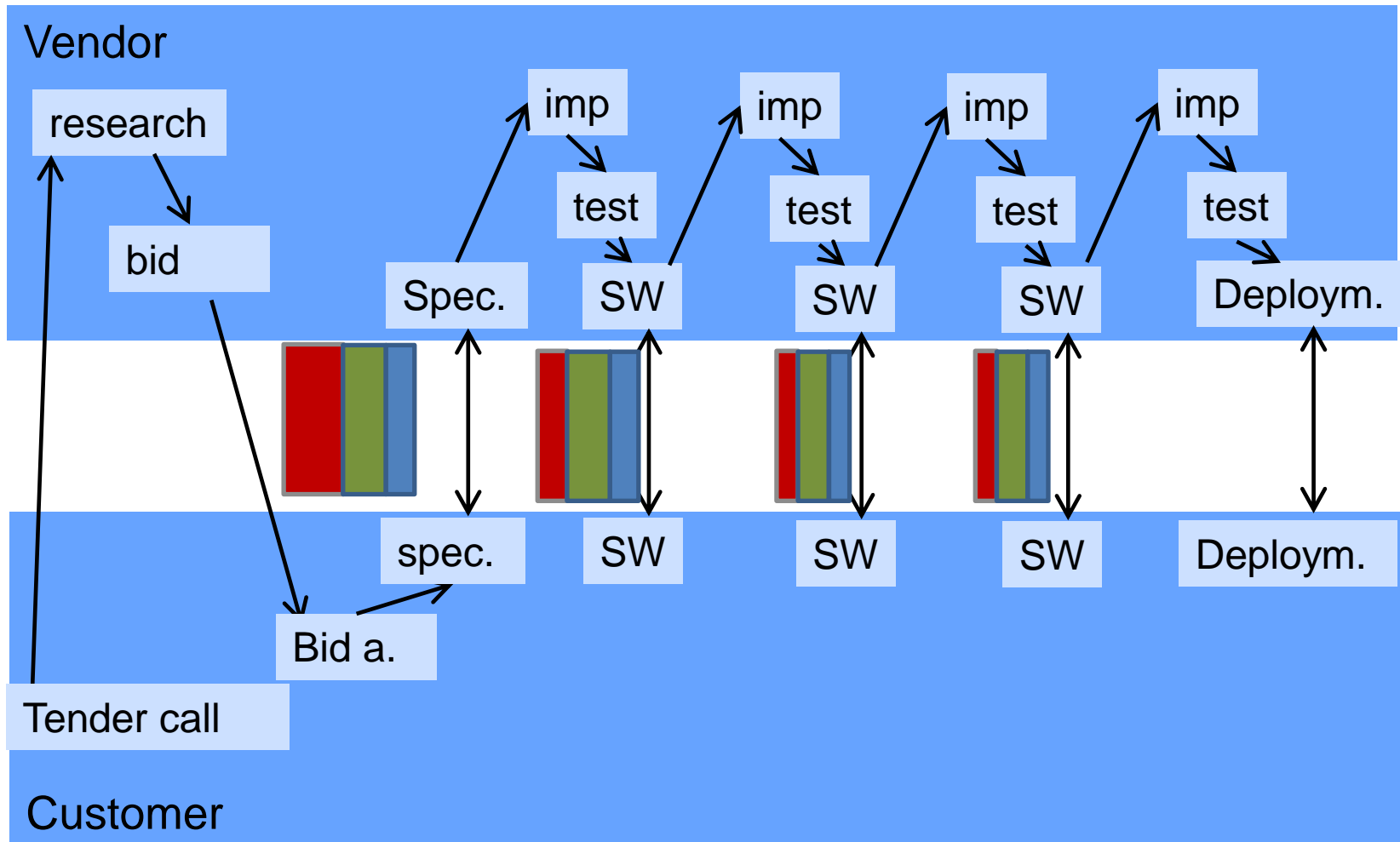
Requirements vs constraints

- **Functional requirement**, for example the software has support for spell checking.
- **Non-functional requirement**, for example User interface follows the UI-guideline of XXX or installation can use at most 5MB disk space.
- **Constraints**, for example the software has to implemented in Windows-operating system in C++-language.

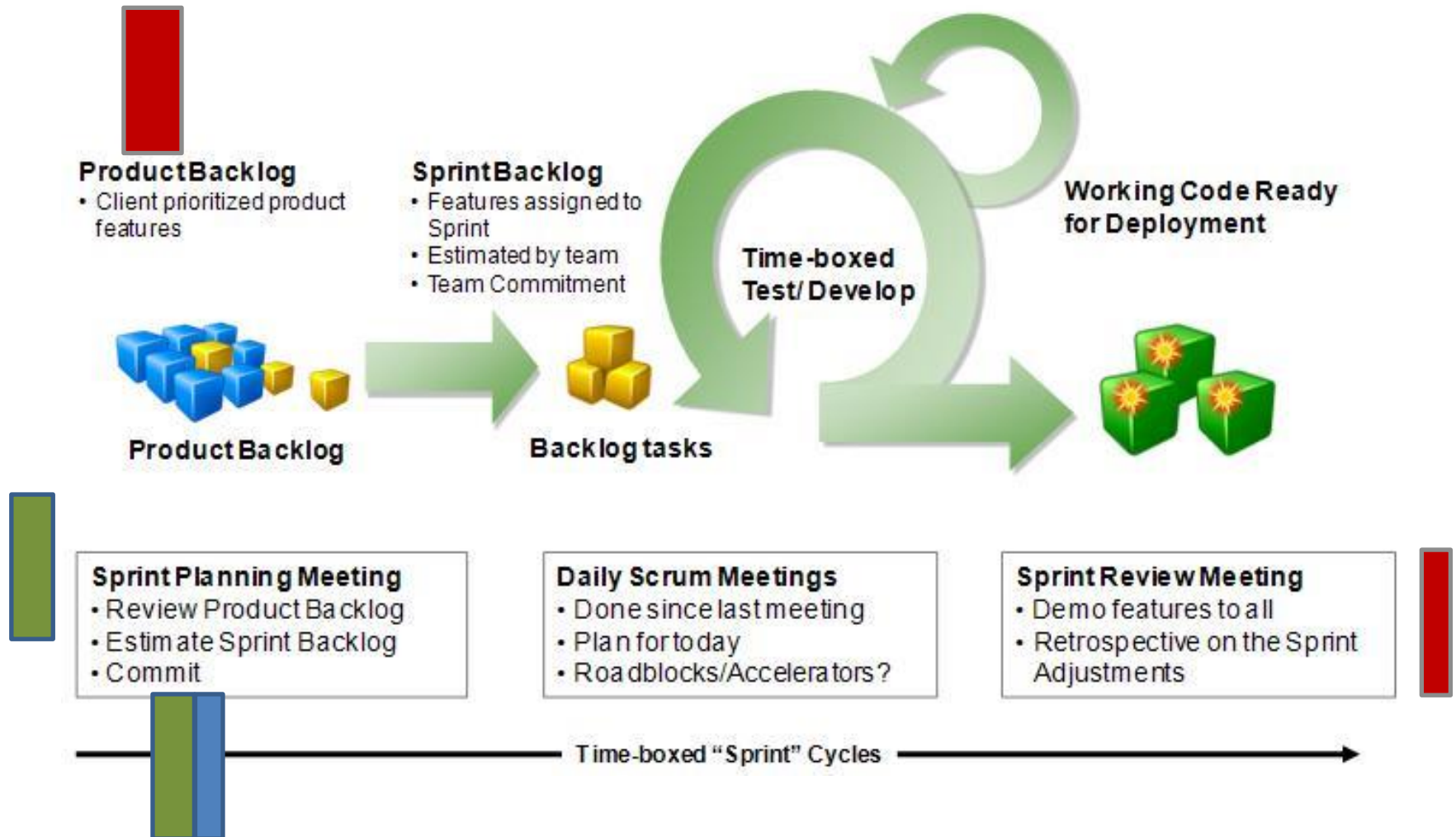
Requirements management



Iterative, agile



Scrum



Requirement errors

(Alan Davis)

- Knowledge errors
 - Requirement is not known
- Triage errors
 - Importance not understood
 - Effort or resources estimated wrong
- Specification errors
 - Documentation missing or not understood

For example

Write the following words in alphabetical order
(the order they come in the alphabet)

A B C D E F G H I J K L M N O P Q R S T U V W X

~~apple~~

pumpkin

log

river

fox

pond

1. apple

2. ikmnpvu

3. log

4. river

5. fox

6. pond

http://en.wikipedia.org/wiki/Requirements_elicitation

In [requirements engineering](#), **requirements elicitation** is the practice of collecting the requirements of a system from users, customers and other stakeholders. ^[1] The practice is also sometimes referred to as **requirements gathering**.

The term elicitation is used in books and research to raise the fact that good requirements can not just be collected from the customer, as would be indicated by the name requirements gathering. Requirements elicitation is non-trivial because you can never be sure you get all requirements from the user and customer by just asking them what the system should do. Requirements elicitation practices include interviews, questionnaires, user observation, workshops, [brainstorming](#), [use cases](#), role playing and [prototyping](#).

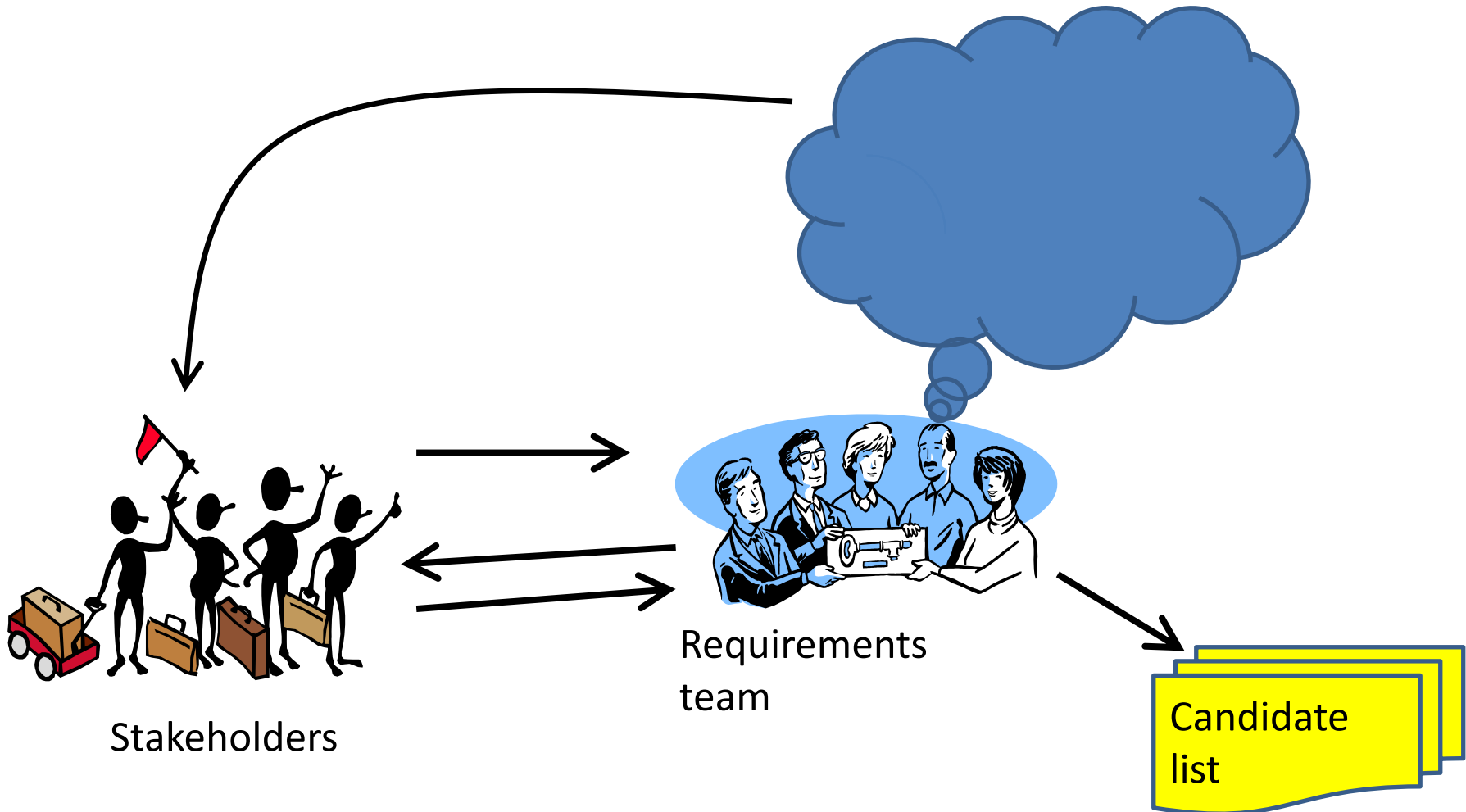
Before requirements can be analyzed, modeled, or specified they must be gathered through an elicitation process. Requirements elicitation is a part of the requirements engineering process, usually followed by analysis and specification of the requirements.

Commonly used elicitation processes are the stakeholder meetings or interviews. For example, an important first meeting could be between software engineers and customers where they discuss their perspective of the requirements.

For a hotel

- I want a telephone system.
 - Why?
- Well, I guess what I really want is a means of communication for all the guests.
 - Why?
- I want our customers be happy.
 - Why?
- I want customers to come again.

From "Just Enough Requirements Management"



User vs system requirements

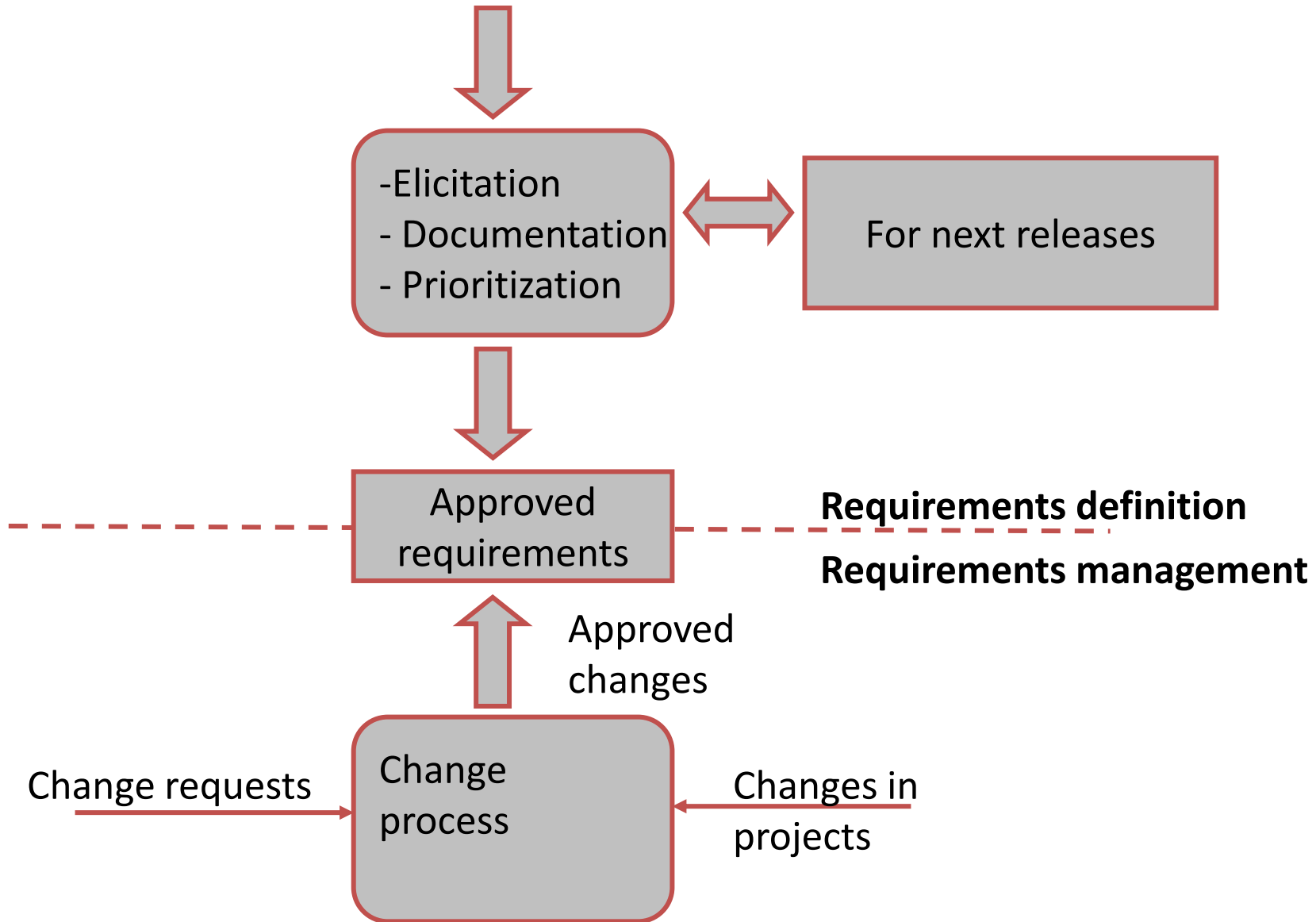
User requirement

- The XXX system shall generate monthly management reports showing the of drugs prescribed by each clinic during that month

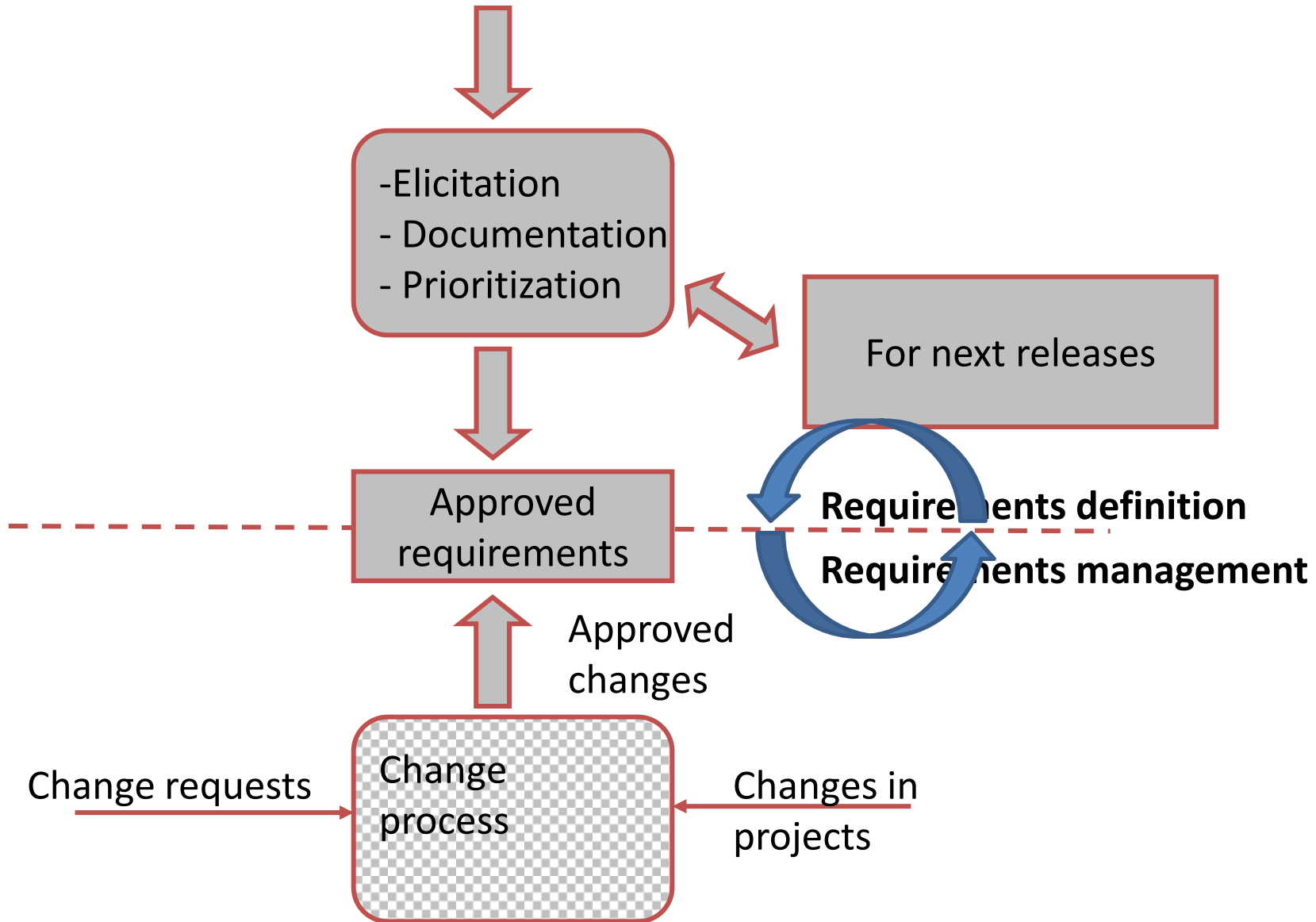
System requirements

- On the last working day of each month a summary of the drugs prescribed, their cost, and the prescribing clinics shall be generated
- The system shall automatically generate the report for printing after 17.30 on the last working day of the month
- A report shall be created for each clinic and shall list the individual drug names, their total prescriptions, the number of doses, and the total cost of the drugs
- If drugs are available n different dose units (e.g. 10mg, 20mg) separate reports shall be created for each dose unit
- Access to all cost reports shall be restricted to authorized users listed on a management access control list

Requirements definition vs management

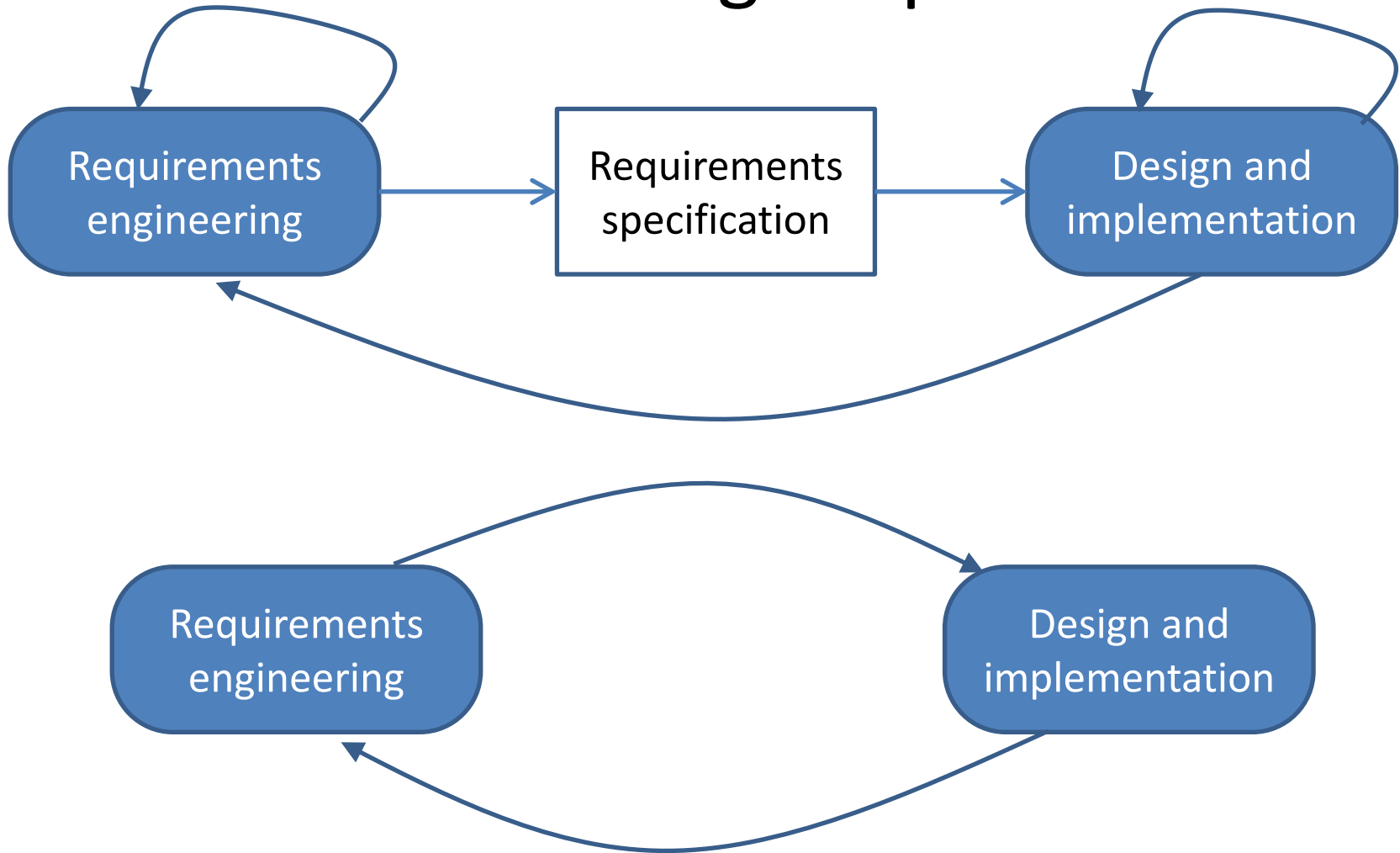


Slightly more agile view



But as Sommerville describes it

Plan-driven vs. agile specification



Techniques and tactics for getting good user requirements

Sources

- Haikala
- Davis
- Sommerville

How to dig customer requirements (developer/vendor viewpoint)

- Nothing replaces knowledge of the application domain
- List stakeholders
 - Think about expectation, wishes, fears, ..
- Discuss with users in their working place
- Plan visit carefully
- Pretend more stupid than you are
- Ask clarification: "you mean that, ...?"
- Use expressions that the customer understand
- Analyze and document each visit, summarize.
- Try to find the original problems
 - Why something has to be done, or is it really necessary?
 - Distinguish essential from old habits
- Prototypes
- User Centered Design

How to tell customer requirements (customer viewpoint)

- Remember: vendor may not understand your business
 - Maybe a selection criteria
- List stakeholders
 - Think about expectation, wishes, fears, ..
- Discuss with users in their working place
- Plan visit carefully
- Pretend more stupid (on information systems) than you are
- Ask clarification: "you mean that, ...?"
- Use expressions that the vendor understand
- Analyze and document each visit, summarize.
- Try to find the original problems
 - Why something has to be done, or is it really necessary?
 - Distinguish essential from old habits
- Prototypes
- User Centered Design

Prototypes

- Makes discussion concrete
- Motivations:
 - Get feedback
 - Ensure that selected technology works
 - Gain commitment
- Evolutionary prototype
 - Stepwise development towards product
- Throw-away prototype
 - Allows optimization

Others

- Observation
- Questionnaires

Properties of a good specification

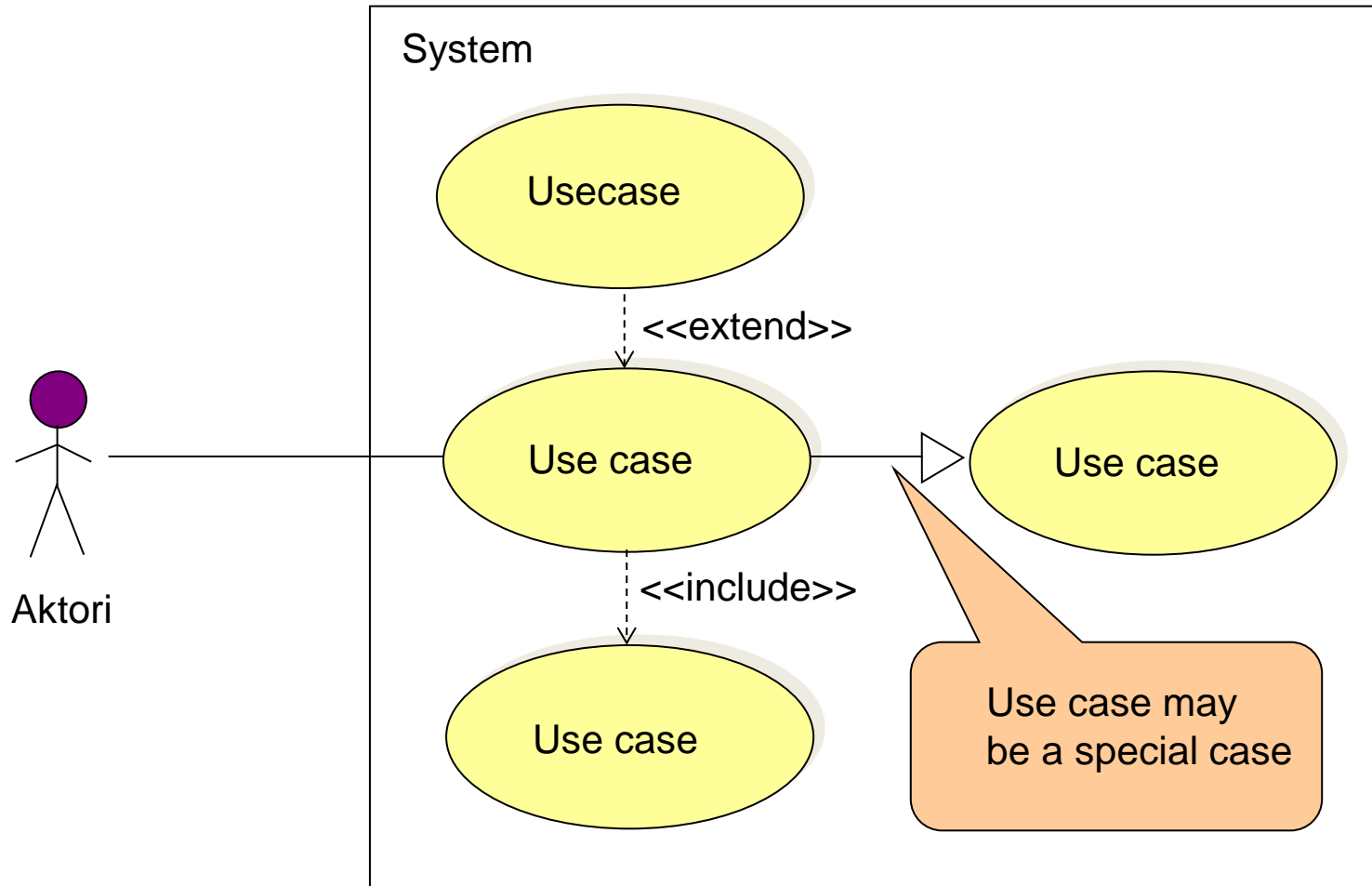
- Complete: all necessary and nothing extra
- Precise
- Error-free
- Understandable
- Testable: is possible to check if requirement has been fulfilled
- Traceability: where the requirement comes and how important it is
- Same topic on one place
- No redundancy (?)

Key points of elicitation

- Never loose sight of the goal: understand enough to avoid risks
- Never think that you understand the problem best
- One stakeholder can never speak on behalf of all
- Maintain glossary of terms
- Avoiding elicitation will make the project longer – not shorter
- Prepare for change
- Accept that all stakeholders have a right to change their mind
- Prepare for triage

UML Use case

(details from TIE-02300)

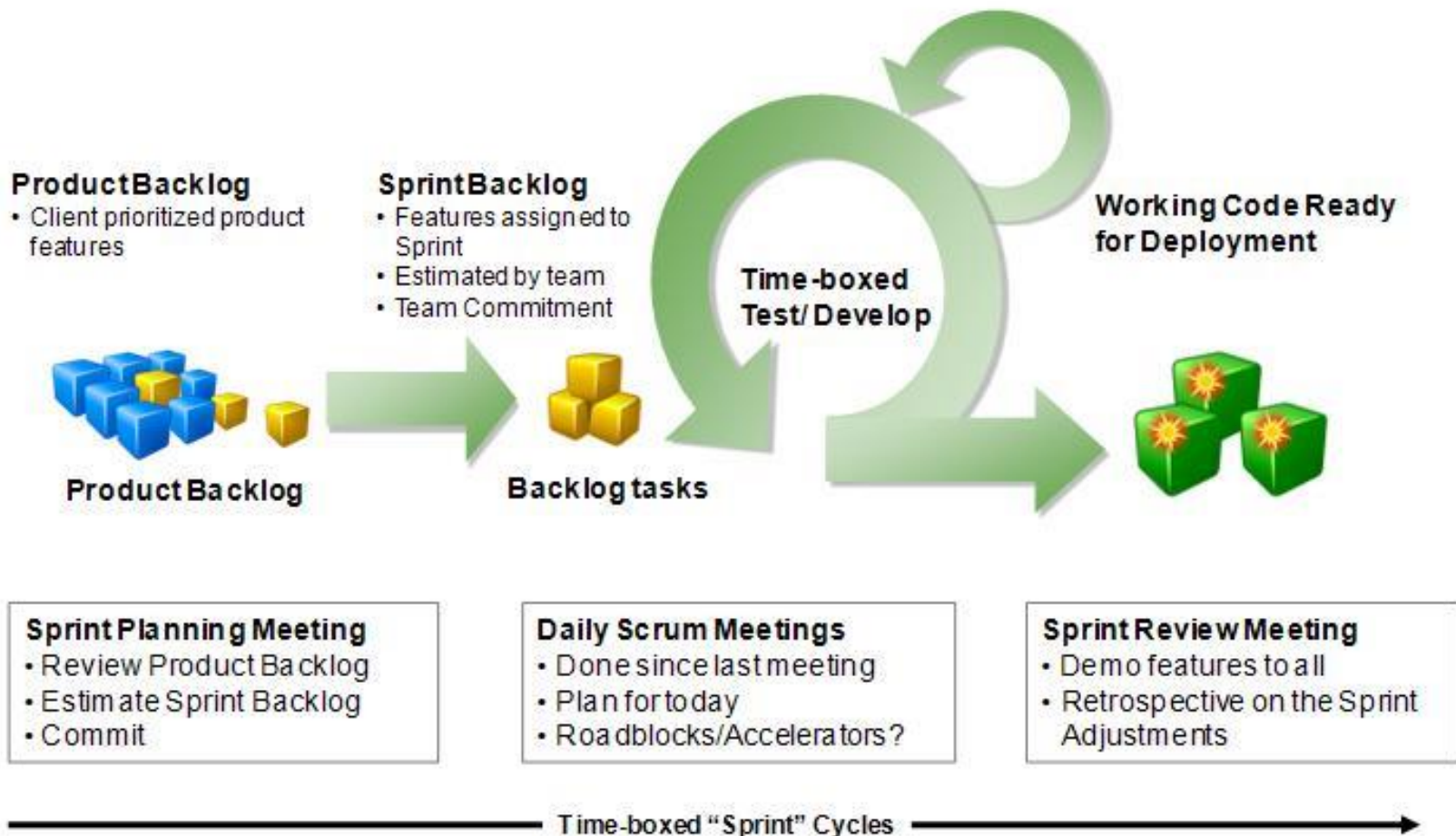


User story

- Used in agile processes
- Simplified use case
- User story lists:
 - Role(s) / Actor(s)
 - What is done
 - Added value of the story (if not self evident)
- Examples
 - As a user, I can backup my entire hard drive.
 - As a power user, I can specify files or folders to backup based on file size, date created and date modified.
 - As a user, I can indicate folders not to backup so that my backup drive isn't filled up with things I don't need saved.

Requirements and Scrum

Scrum

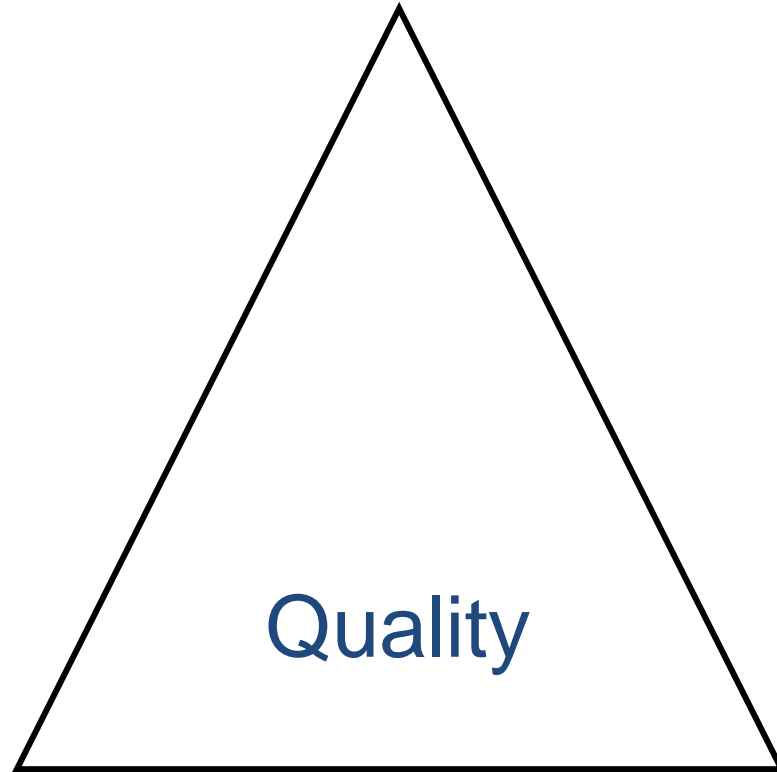


In short

- Although agile prepares for change, initial requirements spec is needed before stating
 - Especially high-level customer requirements
- Product backlog is one kind of living requirement spec

Iron triangle

Scope/features



Quality

Resources

Time/
Schedule

Material

- The Standish Group. The Standish Group Report – CHAOS [WWW]. 1995, <https://cs.nmt.edu/~cs328/reading/Standish.pdf>.