

Software development processes

Life-cycle Models

Lecture 2

Kari Systä

About weekly exercises

- TUE 1015-1200 TC131 Free space
- TUE 1214-1400 TC131 full
- WED 1015-1400 TC163 Free space
- WED 1215-1400 TC131 full
- WED 1415-1600 TC128 full
- WED 1615-1800 TB207 cancelled
- THU 1415-1600 TC163 full
- THU 1415-1600 TC163 Free space

Possible questions to tero.ahtee@tut.fi

About project / assignment

- Will be done in groups of 4
- IDLE for registering will be opened 20.1 and deadline is 27.1.
 - Groups fixed by 31.1
 - If you do not have complete group, you should still register. The staff will combine.
- Project will run in 4 Sprints
- Instructions (still under preparation)
<http://www.cs.tut.fi/kurssit/TIE-21106/assignment/index.html>

For those who are on one of the last courses

- Are you looking for a master thesis topic with ambition to post grad studies after thesis, or are you graduated and want to aim at doctoral studies?
- Are you a skillful SW developer?
- Do you know basics Web and Cloud?
- Interested in working on International ITEA2 project EASI-CLOUDS and collaborate Finnish and European partners.
- In case of 4xYES, please send your application to kari.systa@tut.fi with the following data
 - You CV
 - Study records
 - Initial research plan for your post grad studies
 - Statement on why you want to join the project and what kind of tasks you want to take in the project.

Learning goals of today and the whole course

- What are process models and why they exists?
- Know basics of a few well-known process models
- And what are the motivations behind the models
 - To "behave better"
 - (Some day) to select life-cycle model for your organization
- Know how to participate in the work efficiently

Initial content of lectures

- Introduction
- **Life-cycle models, their background**
- Project management, product management, project planning – in general management aspects
- Scrum in details
- Kanban, Customer Development and DevOps details
- Requirement definition, requirement management, requirements prioritization
- 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

Life-cycle models

<http://www.ics.uci.edu/~wscacchi/Papers/SE-Encyc/Process-Models-SE-Encyc.pdf>

- A software life cycle model is either a descriptive or prescriptive characterization of how software is or should be developed.
- Guideline to organize, plan, staff, budget, schedule and manage software project work over organizational time, space, and computing environments.
- Prescriptive outline for what documents to produce for delivery to client.
- Basis for determining what software engineering tools and methodologies will be most appropriate to support different life cycle activities.
- Framework for analyzing or estimating patterns of resource allocation and consumption during the software life cycle (Boehm 1981)
- Basis for conducting empirical studies to determine what affects software productivity, cost, and overall quality.

Software Process Models

- In contrast to software life cycle models, software process models often represent a networked sequence of activities, objects, transformations, and events that embody strategies for accomplishing software evolution.
- Such models can be used to develop more precise and formalized descriptions of software life cycle activities.

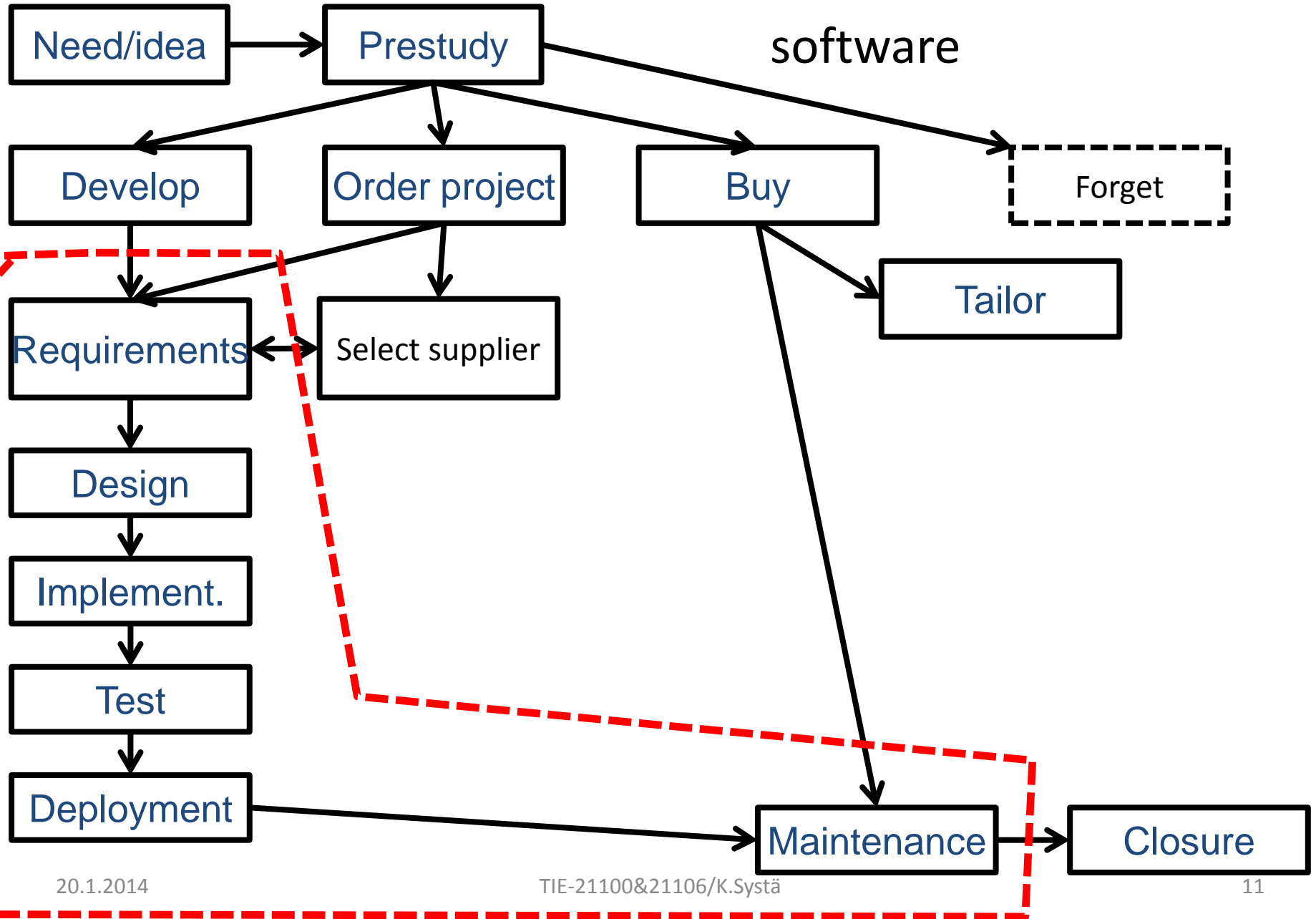
Life-cycle model and process model

- Often seen as synonyms. For example wikipedia.org/wiki/Software_development_process writes:
 - A software development process, also known as a software development life-cycle (SDLC), is a structure imposed on the development of a software product.
- https://ece.uwaterloo.ca/~se464/06ST/lecture/02_life-cycle-models.pdf:
- Lifecycle models: Phases in the life of an artifact, e.g., a system
- Process models: Activities performed on artifacts, e.g.,
- development activities

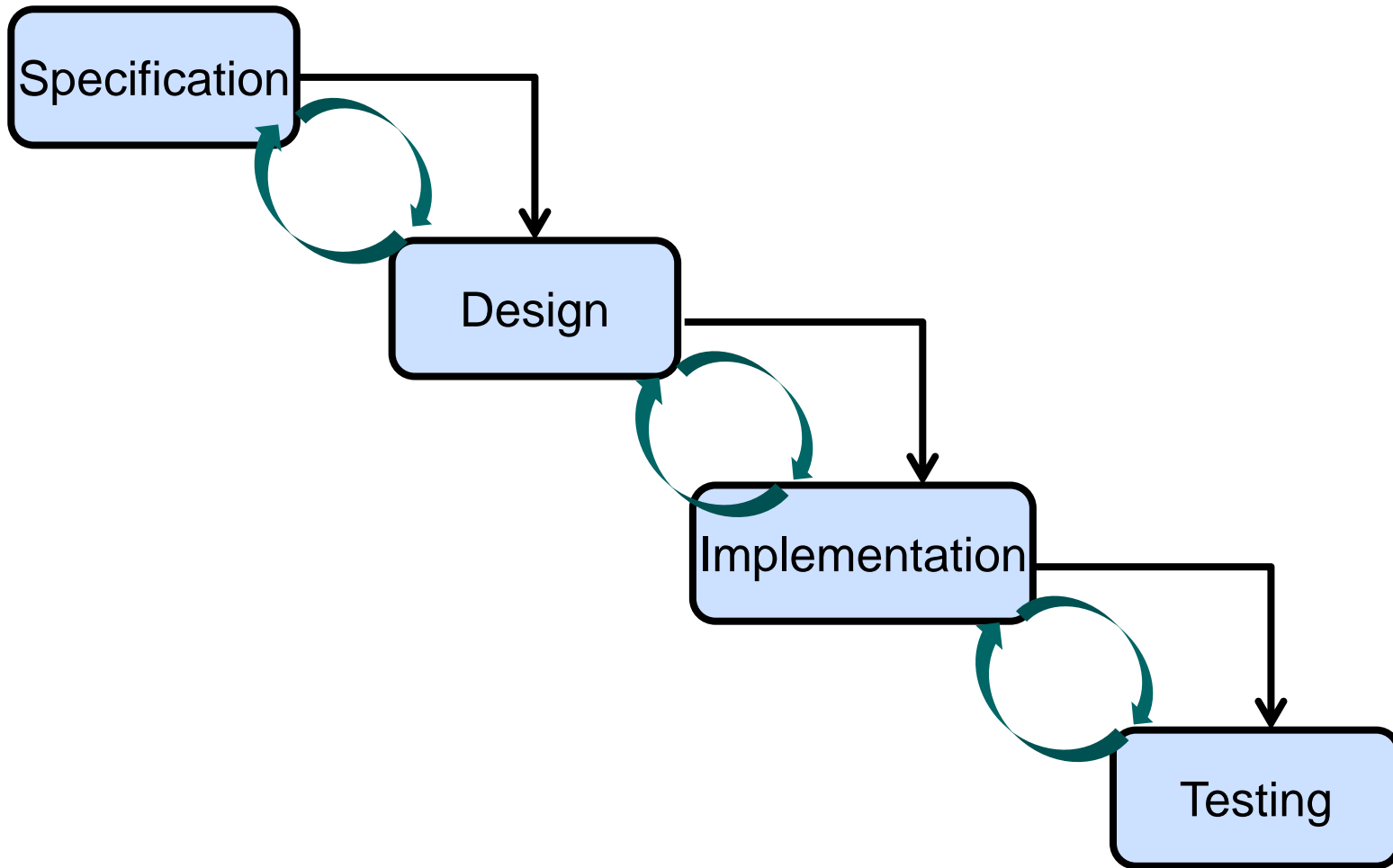
Don't PANIC

- We mostly do not care about the possible difference in this course

From needs to
software

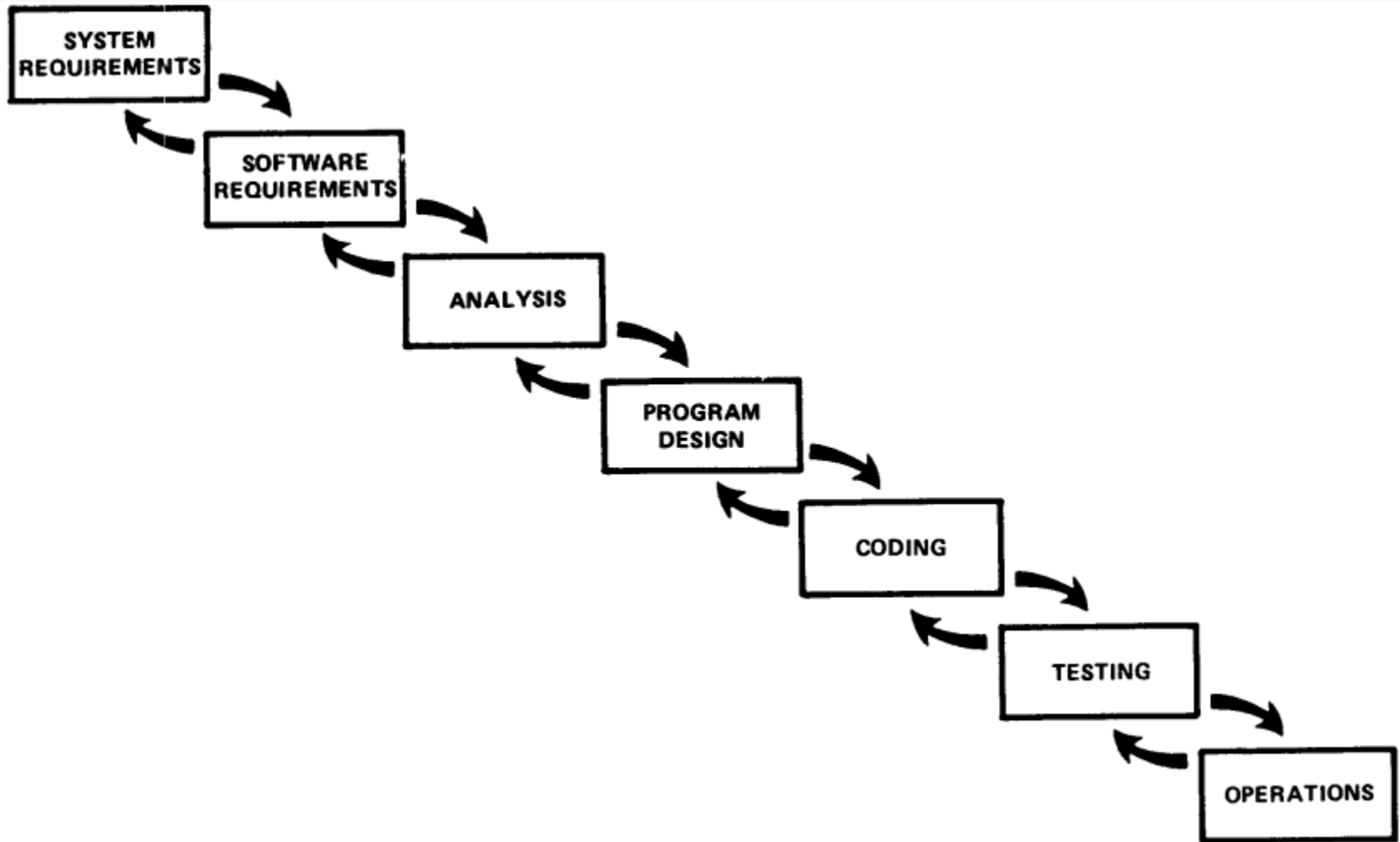


Waterfall model - simplified



Royce, 1970

(<http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf>)

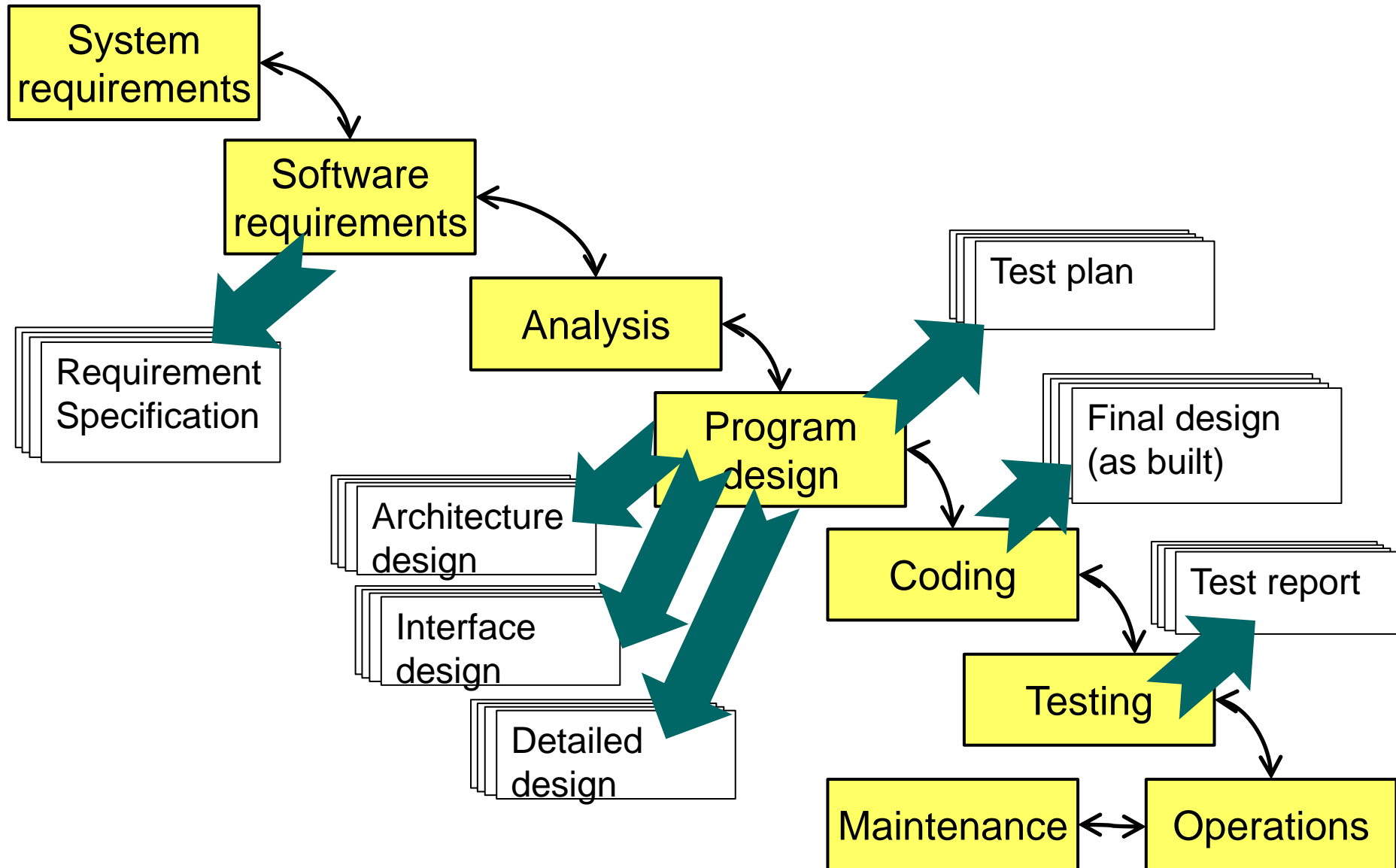


Principles of waterfall

- Waterfall is often understood as one-directional flow, but Royce considered iterations as a crucial part of the model.
(mainly between consecutive steps)
- Waterfall is a plan-driven approach
- Move from step to next is a decision and often involves reviews, re-planing, budget decisions etc.
- Proper design and plan prevents extra work (and cost) in next steps
- The earlier the mistake is done, the more expensive it is
- Waterfall is consistent with other engineering processes

Documentation is a crucial part of waterfall

(on possible example)



Old and classic joke



How the customer explained it



How the Project Leader understood it



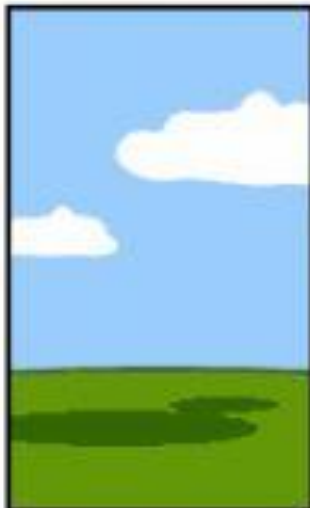
How the Analyst designed it



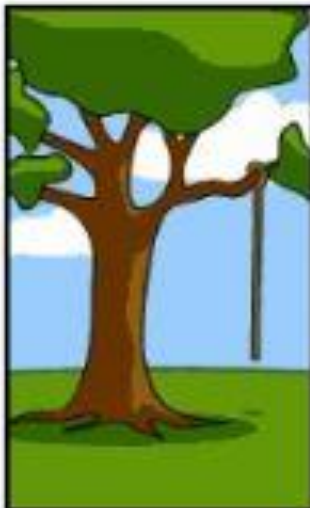
How the Programmer wrote it



How the Business Consultant described it



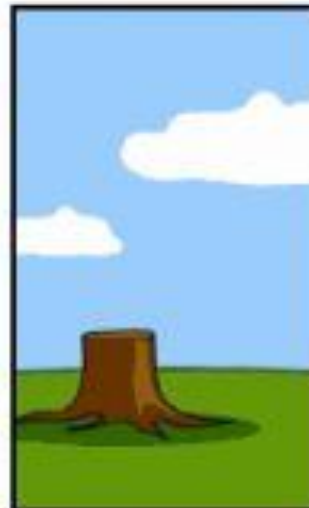
How the project was documented



What operations installed



How the customer was billed



How it was supported



What the customer really needed

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,

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

Prototyping

- Motivations:
 - Get feedback
 - Ensure that selected technology works
 - Gain commitment
- Evolutionary prototype
 - Stepwise development towards product
- Throw-away prototype
 - Allows optimization
- Can be combined with waterfall

Precursor of interative models: Spiral Model

(picture from: <http://www.sei.cmu.edu/reports/00sr008.pdf>)

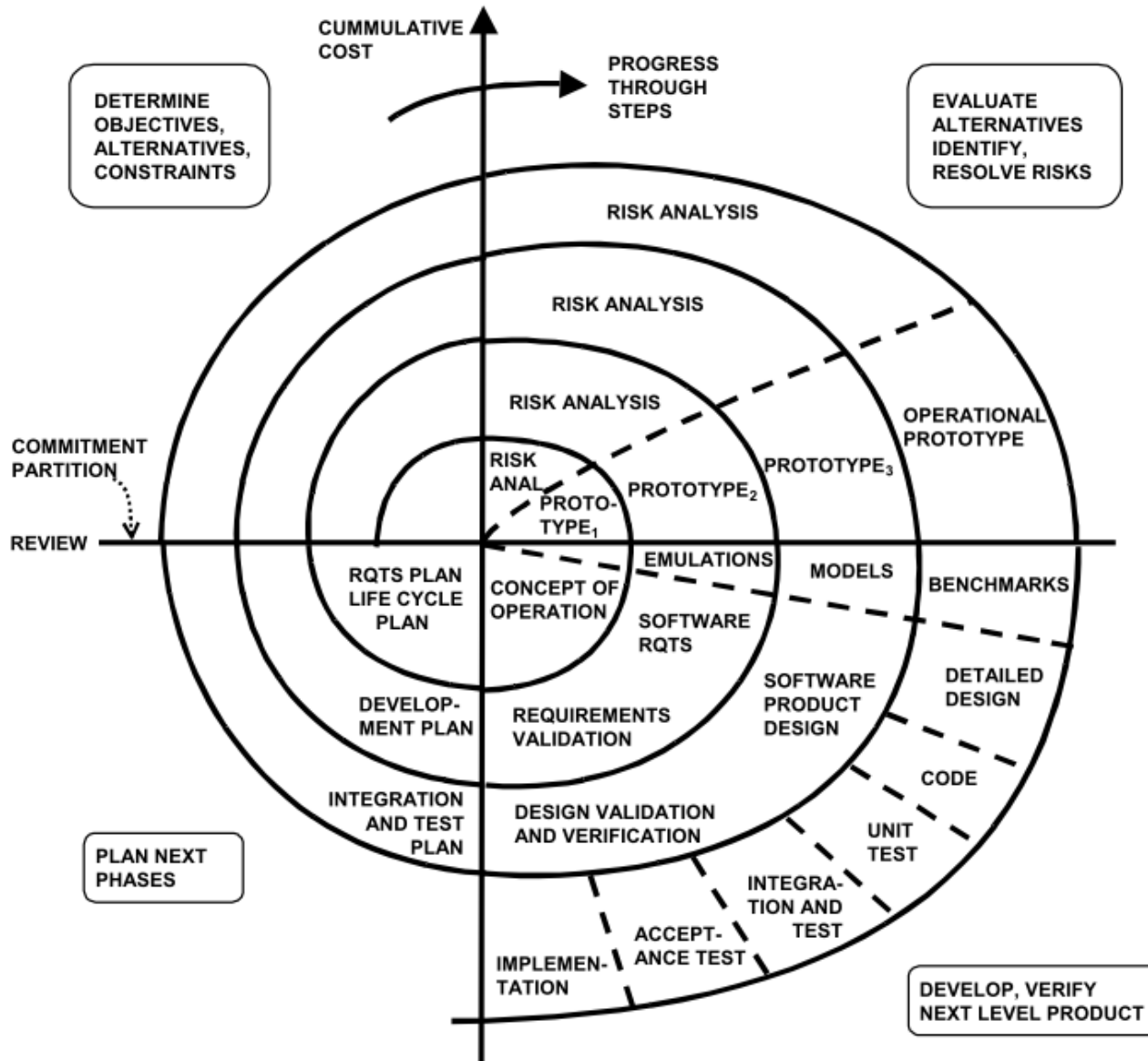


Figure 1: Original Diagram of Spiral Development

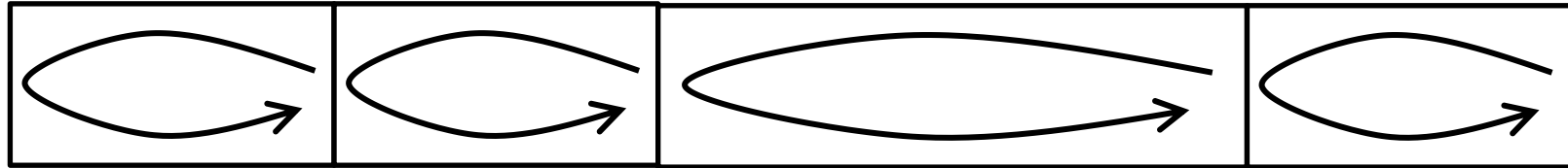
Fundamentals of spiral model

- Spiral model is a risk-driven process and handling risks is explicit
- Each loop is split to four sectors
 - 1) Objective setting
 - 2) Risk assessment and reduction
 - 3) Development and validation
 - 4) Planning
- Spiral model is not a series of waterfalls !
(A common misconception)

Rational Unified Process (RUP)

- Derived from UML and Unified Software Development Process
- Large and complex, but the purpose is not that all companies adopt all practices
- Three views
 - Dynamic: phases over time
 - Static: process activities
 - Practice

Phases in RUP



Inception

Elaboration

Construction

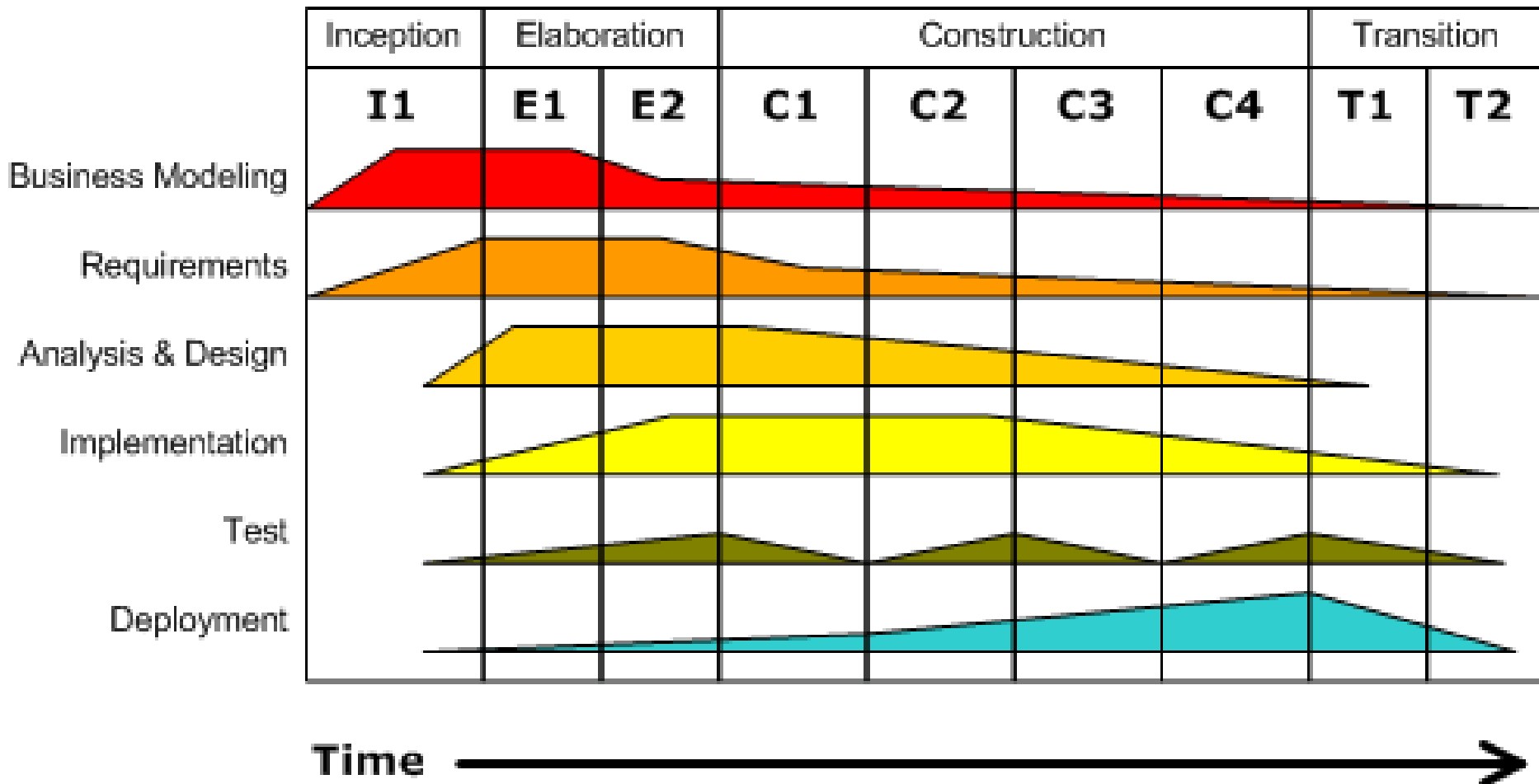
Transition

- Inception: business case and stakeholders
- Elaboration: spec, design, plan
- Construction: "the real work"
- Transition: to users

Iterative model: RUP (Rational Unified Process)

Iterative Development

Business value is delivered incrementally in time-boxed cross-discipline iterations.



Key points so far

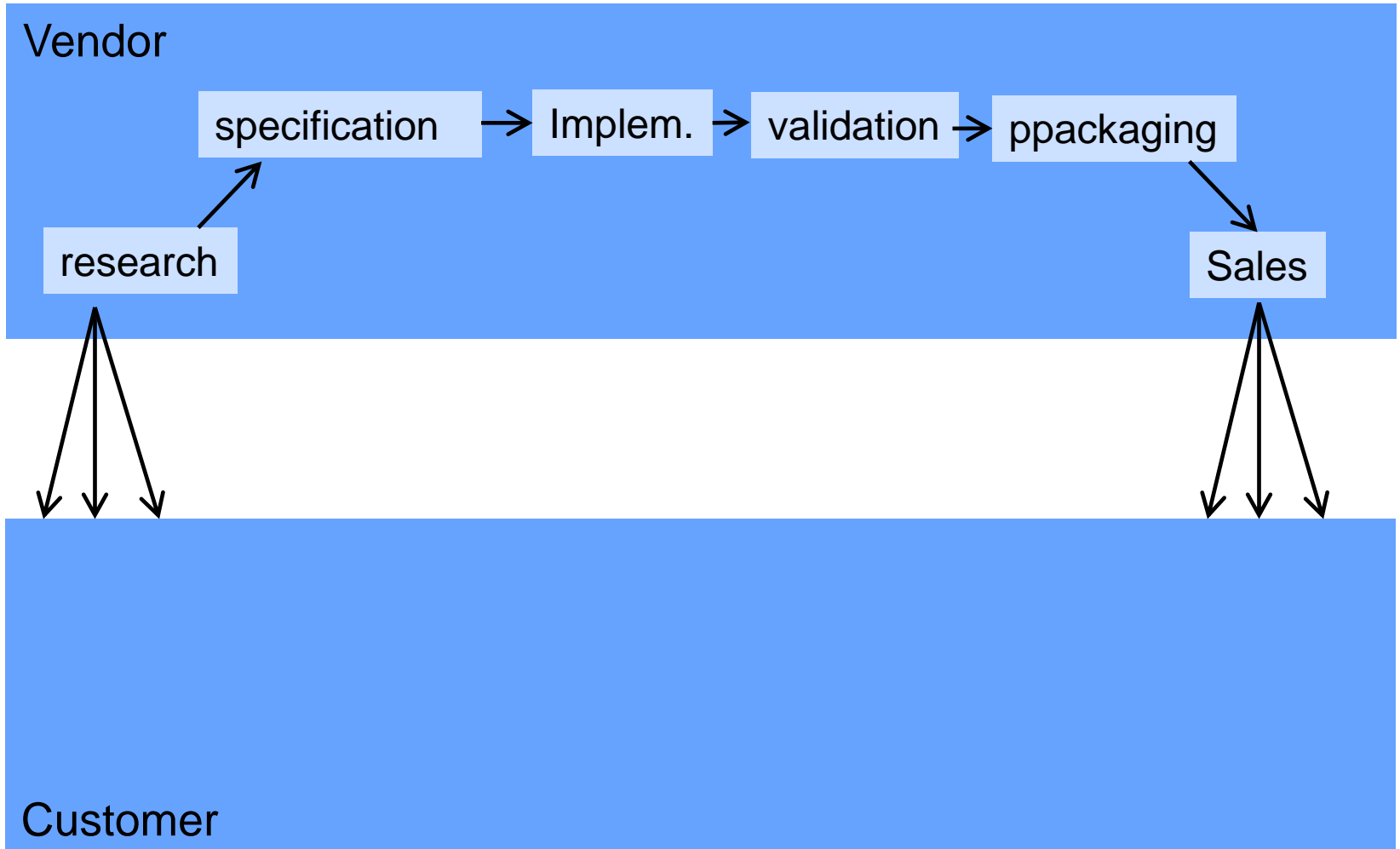
- Software processes is set of activities in SW production; software process model is abstract representation
- General process models describe organization of the process
- Requirements engineering develops software specification
- Design and implementation transfer specification to executable software
- Validation checks that confirms specification and real user requirements
- Evolution changes software to meet new requirements
- Process should include activities to cope with the change

Now a break and then

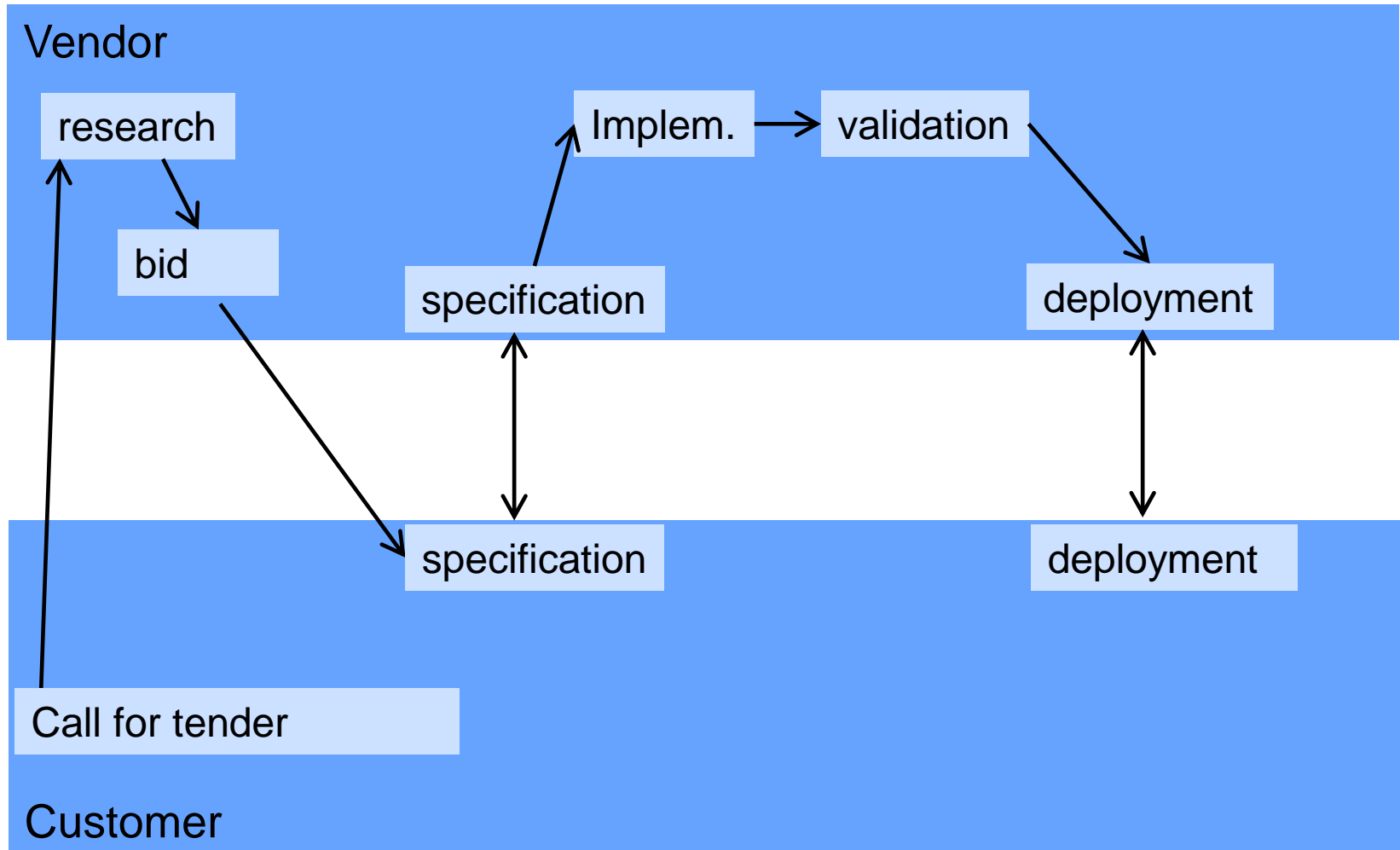
- we put the previous discussion into context
- introduction to agile



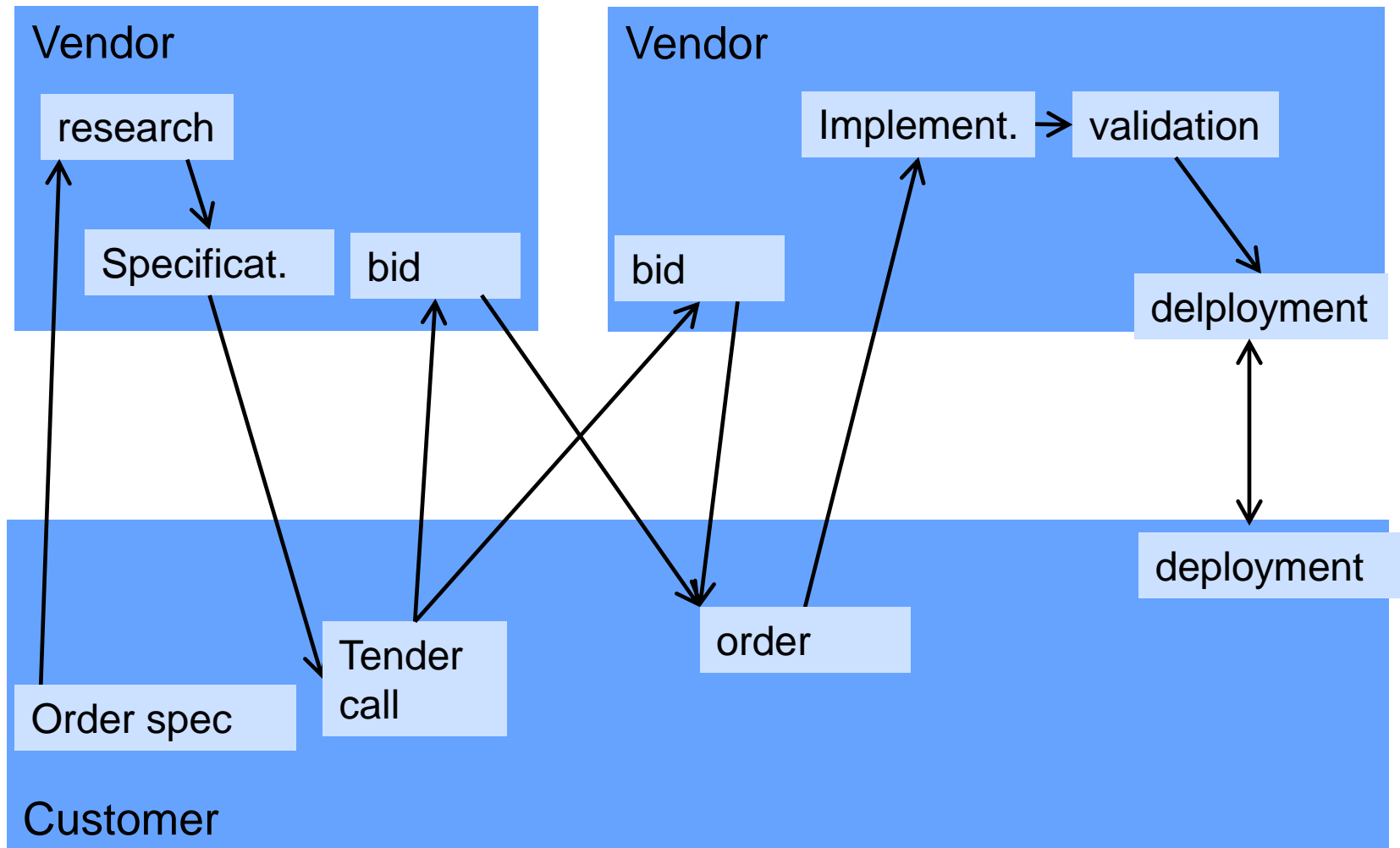
Different projects - product



Different project – customer specific



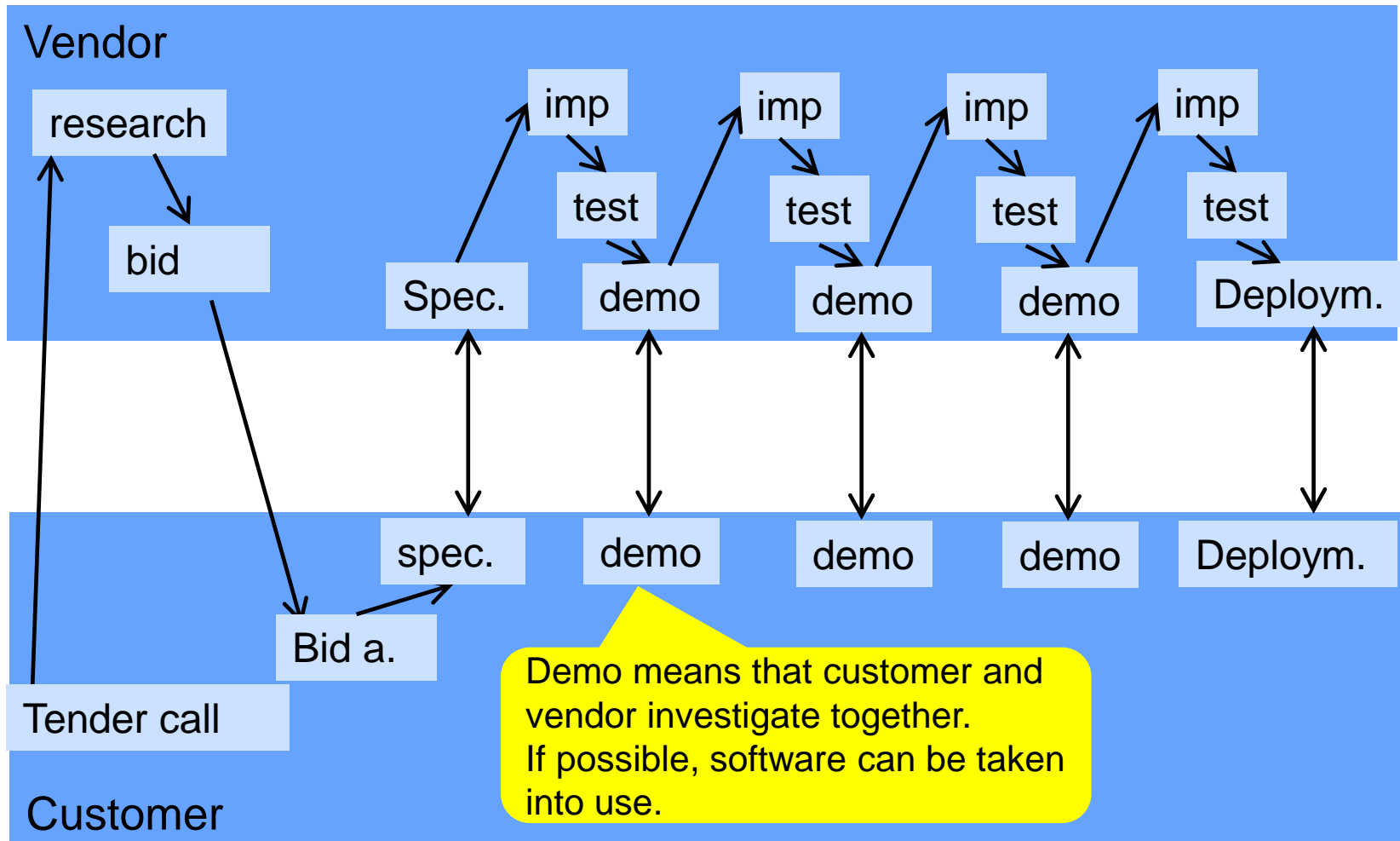
Different project – customer specific



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 glueing components together
 - But: the components are implemented by humans
- Assumption 4: schedule is followed
 - Actually very seldom

Iterative, agile



Agile -manifest

- February 2001
- 17 "inventors"
- We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:
 - **Individuals and interactions** over processes and tools
 - **Working software** over comprehensive documentation
 - **Customer collaboration** over contract negotiation
 - **Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Manifesti Suomeksi

Me etsimme parempia keinoja ohjelmistojen kehittämiseen tekemällä sitä itse ja auttamalla siinä muita. Tässä työssämme olemme päätyneet arvostamaan

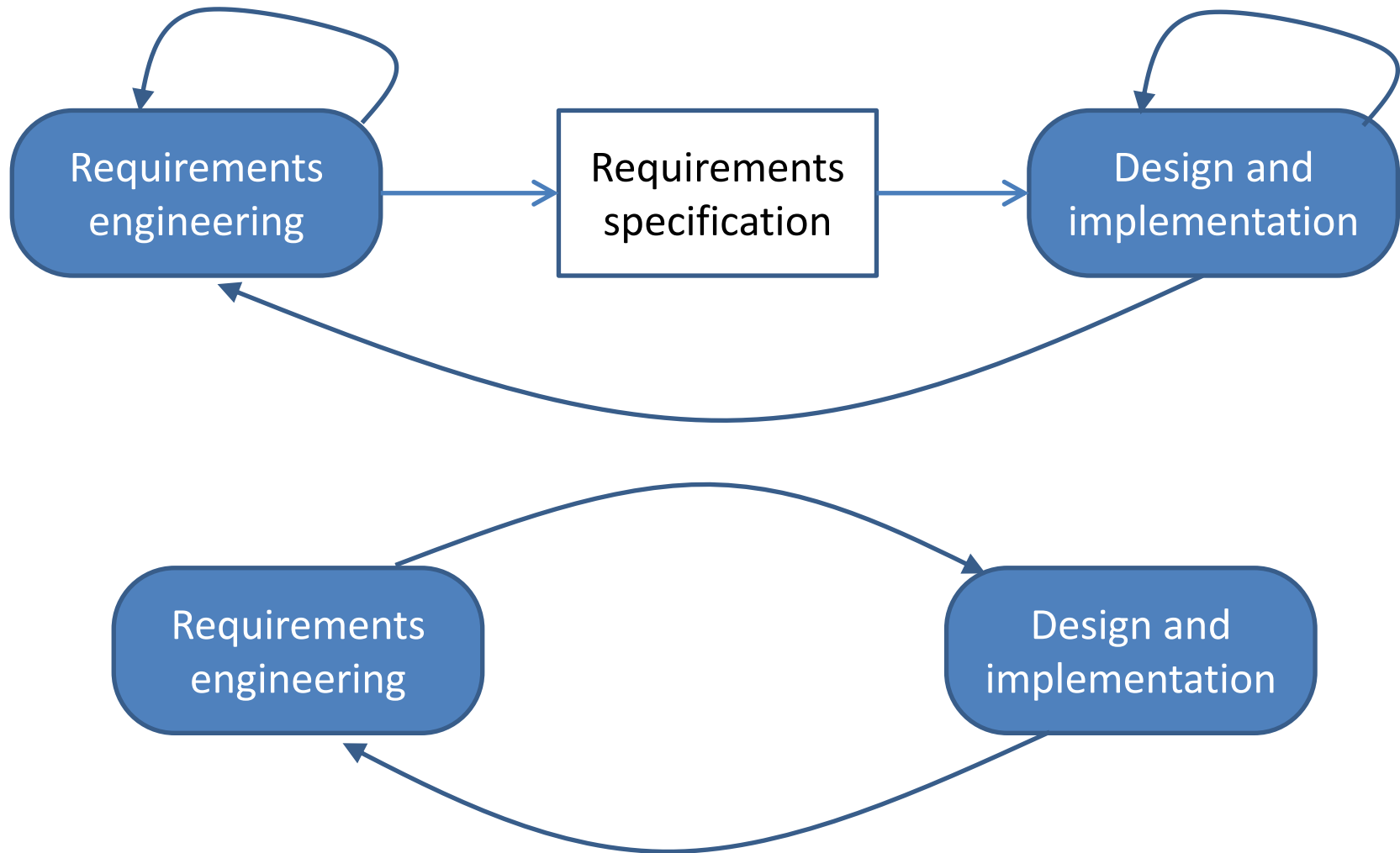
- **Yksilöitä ja vuorovaikutusta** enemmän kuin prosesseja ja työkaluja
- **Toimivaa sovellusta** enemmän kuin kokonaisvaltaista dokumentaatiota
- **Asiakasyhteistyötä** enemmän kuin sopimusneuvotteluita
- **Muutokseen reagoimista** enemmän kuin suunnitelman noudattamista.

Vaikka oikeallakin puolella on arvoa, me arvostamme vasemmalla olevia asioita enemmän.

Five principles of Agile

Customer involvement	Through the project. Provide and prioritize requirements, evaluate iterations
Incremental delivery	Customer specifies the increments
People not process	Skill recognized and exploited; Team should decide on ways of working
Embrace change	Plan and design for change
Maintain simplicity	Both in process and software

Plan-driven vs. agile specification



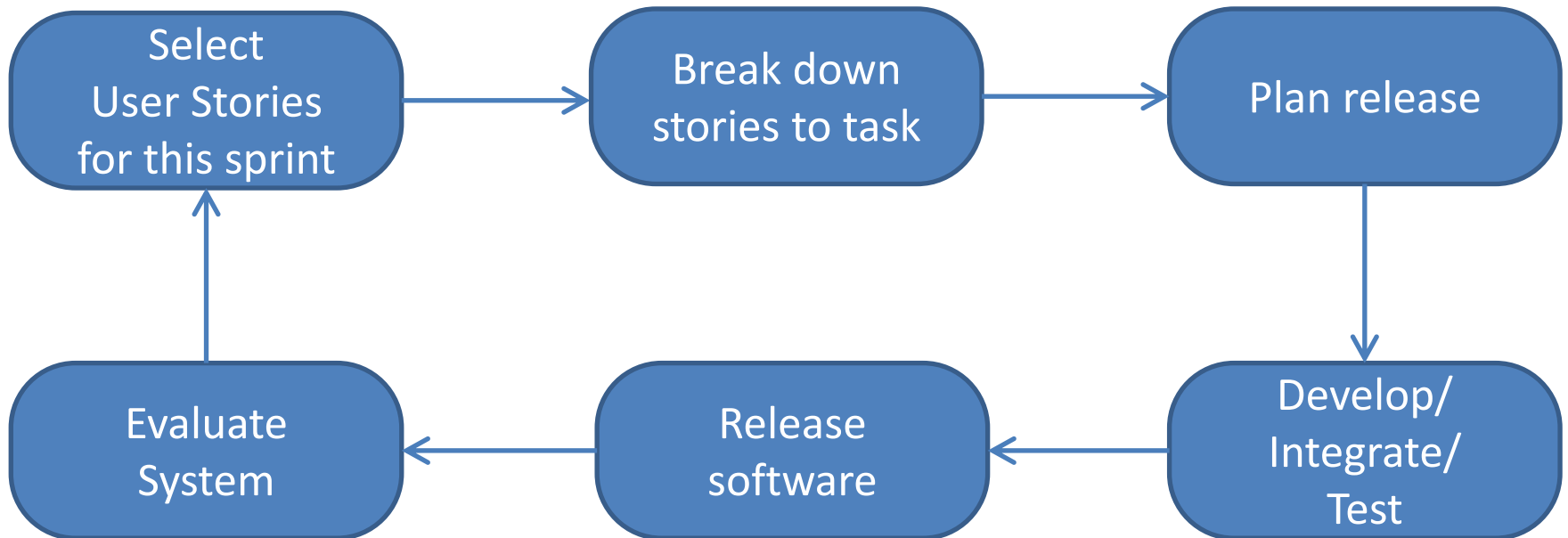
Problems in realizing Agile

- Getting customer commitment and trust is difficult
- Some team members do not have suitable personalities
- Prioritizing changes is difficult – especially if there are many stakeholders
- Maintaining simplicity requires extra work
- Cultural changes through the company
(Agile training should start from management)

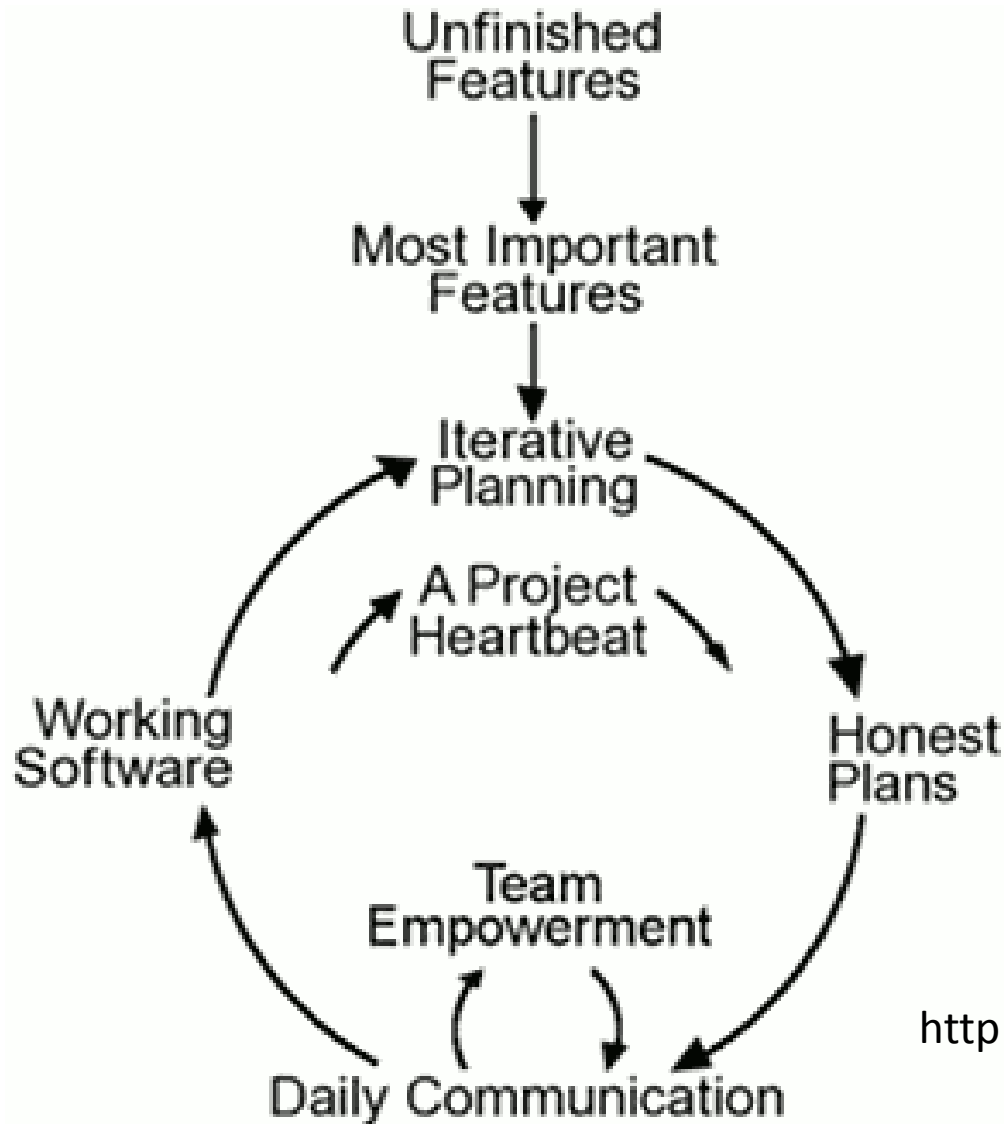
Extreme programming

- Well-known method developed by Kent Beck
- (only briefly covered by Haikala&Mikkonen, but internet is full of resources)
- Requirements are expressed as scenarios called *User Stories*
- which are implemented directly as series of tasks
- Programmers work in pairs and develop tests for each task before writing the code
- All tests must be successfully executed when new code is integrated

XP release cycle



XP (Extreme programming)



<http://www.extremeprogramming.org/>

XP Practices

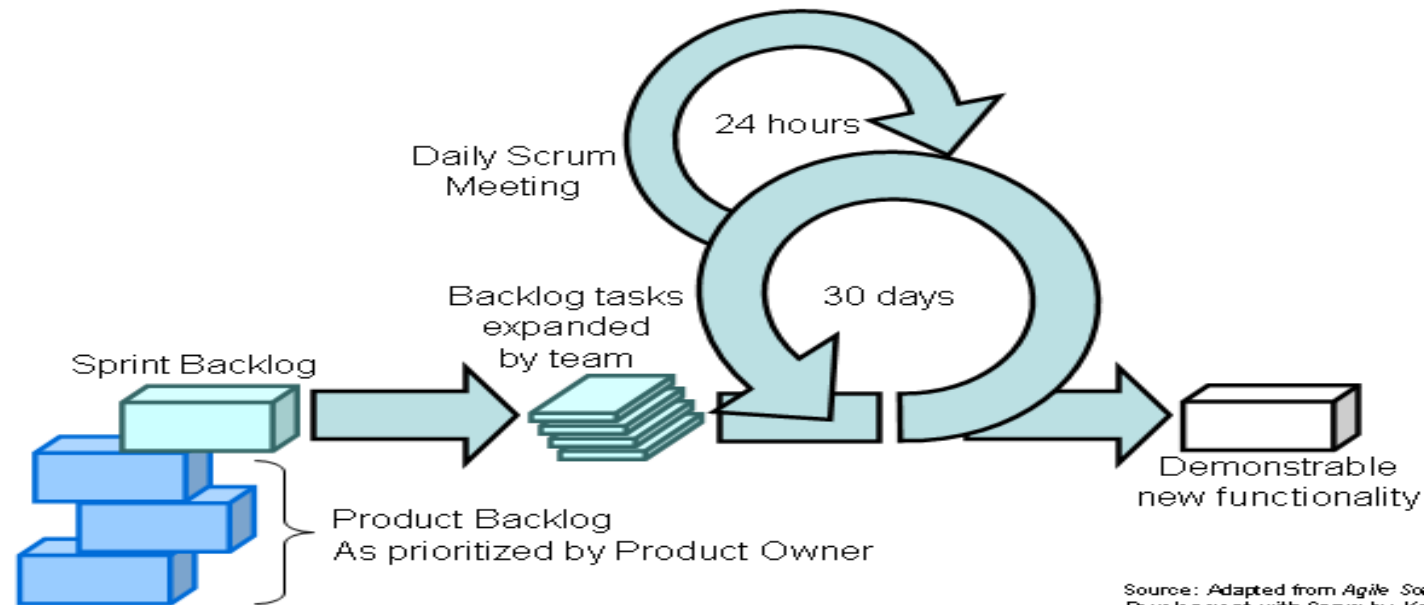
Practice/principle	Description
Incremental planning	
Small releases	Minimum useful is implemented first; frequent releases
Simple design	Spend enough time in design
Test-driven development	Test written before code, automated tests
Refactoring	All team members should refactor code to keep simple and maintainable
Pair programming	Check each others' work; support
Collective ownership	No islands of responsibilities; every body can change everything
Continuous integration	Whenever something is ready it is integrated; always test
Sustainable pace	Large amounts of overtime is not sustainable
On-site customer	Continuous access to customer

About pair-programming

- Supports collective ownership and responsibility
- Informal review because each line of code has been seen more than one person
- Supports refactoring
- Fosters learning from colleagues
- Research on productivity gives mixed results

Scrum

- Framework for agile and iterative development
- Jeff Sutherland, John Scumniotales, and Jeff McKenna OOPSLA 95



Source: Adapted from *Agile Software Development with Scrum* by Ken Schwaber and Mike Beedle.

Scrum-rooles

- Pigs
 - Scrum master
 - Product owner
 - Team members
- Chicken
 - Stakeholders (customer,...)
 - Managers



XP vs Scrum

- XP has typically shorter iterations (1-2w instead of 2-4w)
- Scrum does not allow changes into sprints
- XP is work in strict priority order
- Scrum does not prescribe any engineering practices
- Scrum focuses more on management aspects

Learning goals of today and the whole course

- What are process models and why they exists?
- Know basics of a few well-known process models
- And what are the motivations behind the models
 - To "behave better"
 - (Some day) to select life-cycle model for your organization
- Know how to participate in the work efficiently

Initial content of lectures

- Introduction
- **Life-cycle models, their background**
- Project management, product management, project planning – in general management aspects
- Scrum in details
- Kanban, Customer Development and DevOps details
- Requirement definition, requirement management, requirements prioritization
- 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

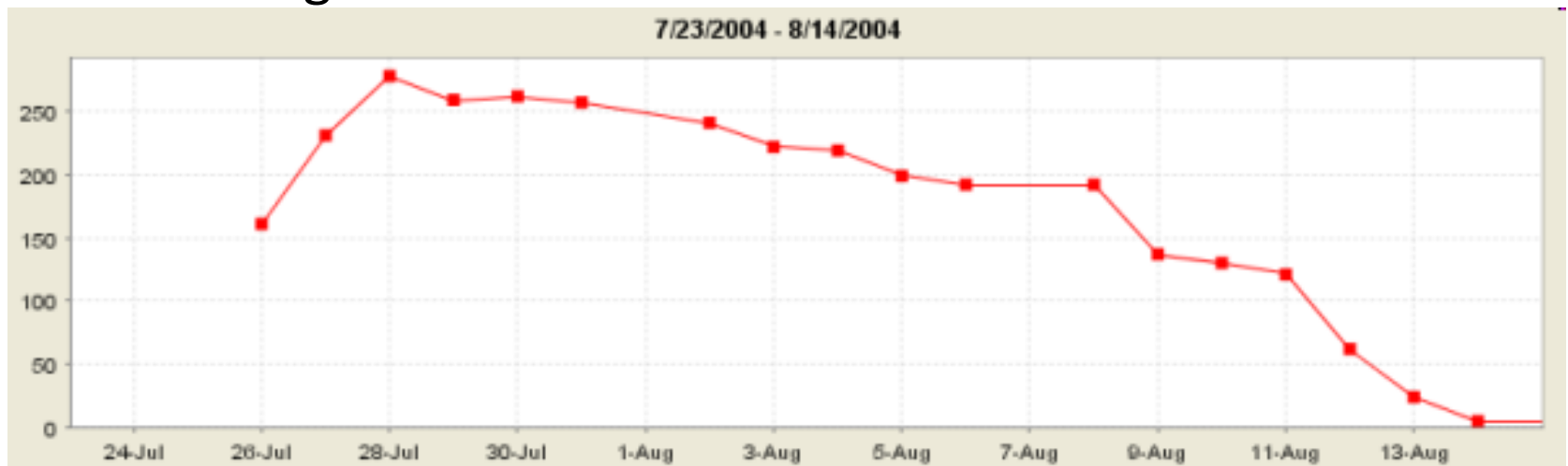
Links and further reading

- Managing the Development of Large Software Systems(Royce, 1970)
<http://www.cs.umd.edu/class/spring2003/cmssc838p/Process/waterfall.pdf>
- Walt Scacchi, Process Models in Software Engineering, in Encyclopedia of Software Engineering, 2 nd Edition, John Wiley and Sons, Inc, Dec 2001.
<http://www.ics.uci.edu/~wscacchi/Papers/SE-Encyc/Process-Models-SE-Encyc.pdf>
- “Competing” lecture:
https://ece.uwaterloo.ca/~se464/06ST/lecture/02_life-cycle-models.pdf
- Spiral model revisited by Barry Boehm him self:
<http://www.sei.cmu.edu/reports/00sr008.pdf>
- RUP best practices by IBM:
http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf
- Lot of good material about agile methods:
<http://www.agilealliance.org/resources/>

The following slides postponed
to next lecture

Burndown -Chart

- Done -> 100% done
- Velocity -> The velocity is calculated by counting the number of units of work completed in a certain interval (sprint in case of Scrum)
- When task is done value in chart is reduced
- If the task grows value is increased



Timeboxing

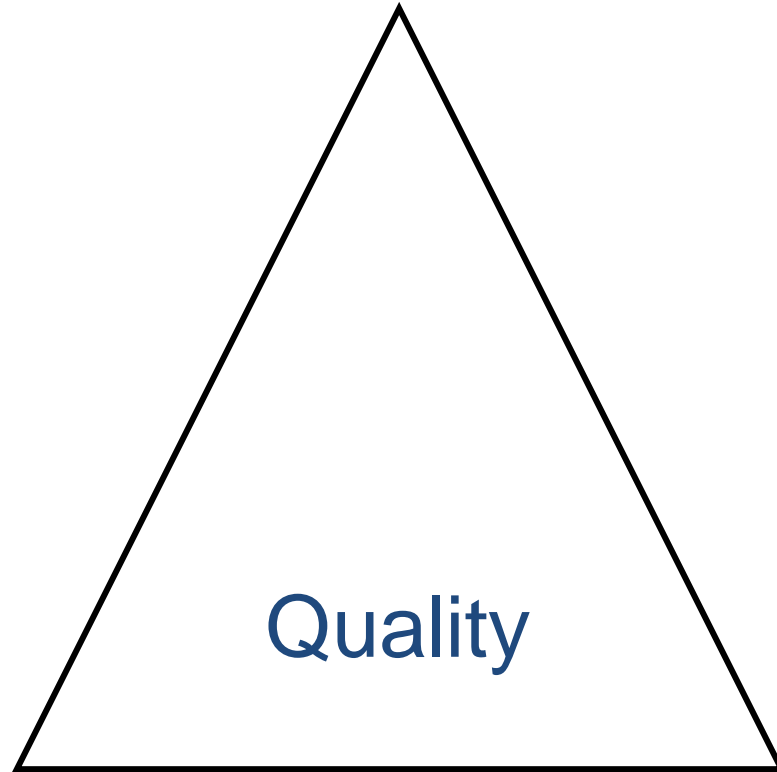
- <http://guide.agilealliance.org/guide/timebox.html>:
 - A timebox is a previously agreed period of time during which a person or a team works steadily towards completion of some goal. Rather than allow work to continue until the goal is reached, and evaluating the time taken, the timebox approach consists of stopping work when the time limit is reached and evaluating what was accomplished

TB1	TB2	TB3
-----	-----	-----

- Ways to manage risks
- Fast response
- Makes requirement management more rigid

Iron triangle

Scope/features



Quality

Resources

Time/
Schedule