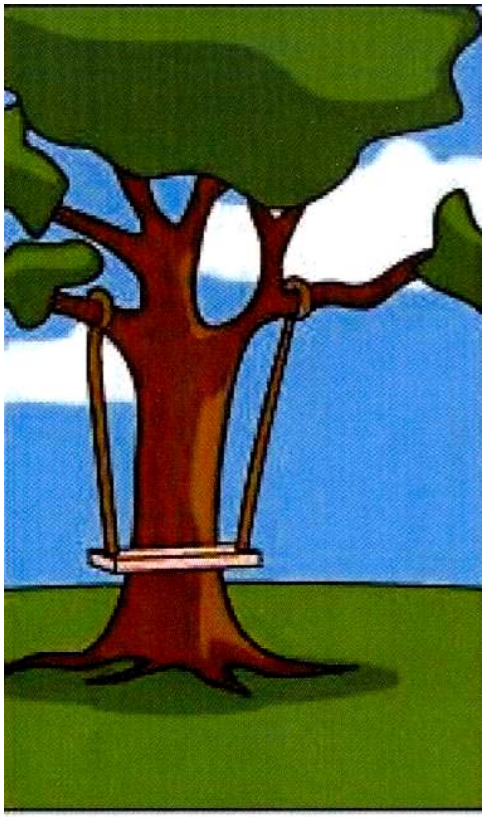
LISA: Sissejuhatus TARKVARATEHNIKASSE

2006

Old Pictures

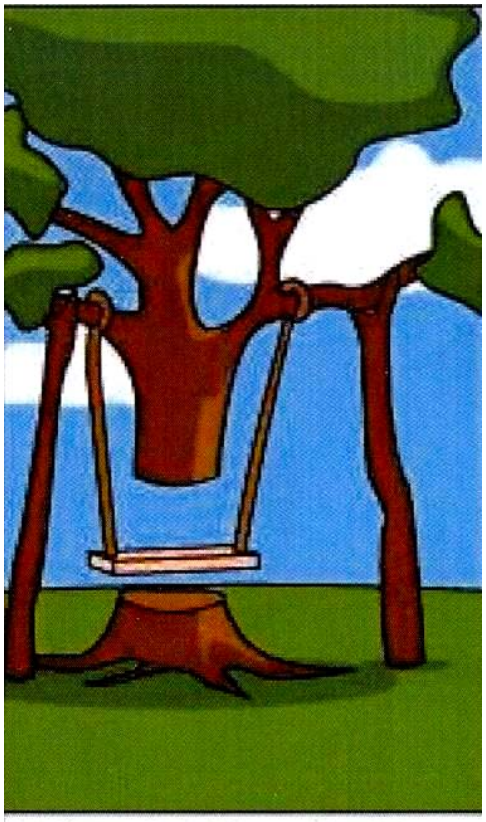
How the customer explained it

How the customer explained it



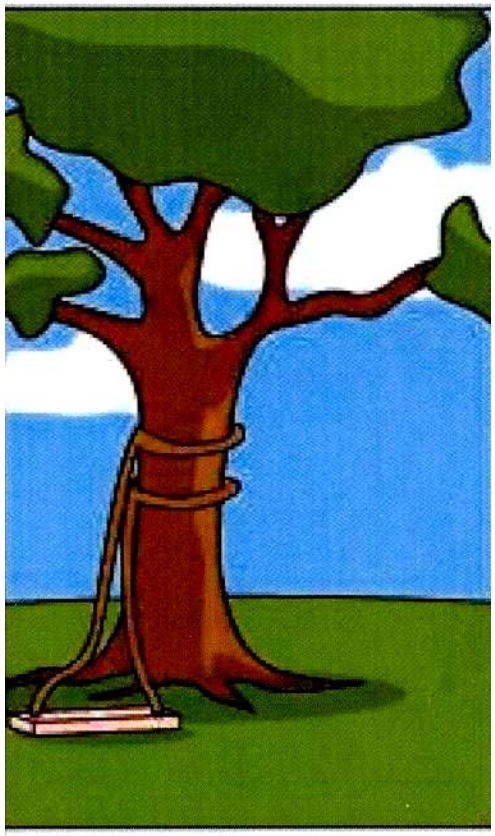
How the Project Leader understood it

How the Project Leader understood it



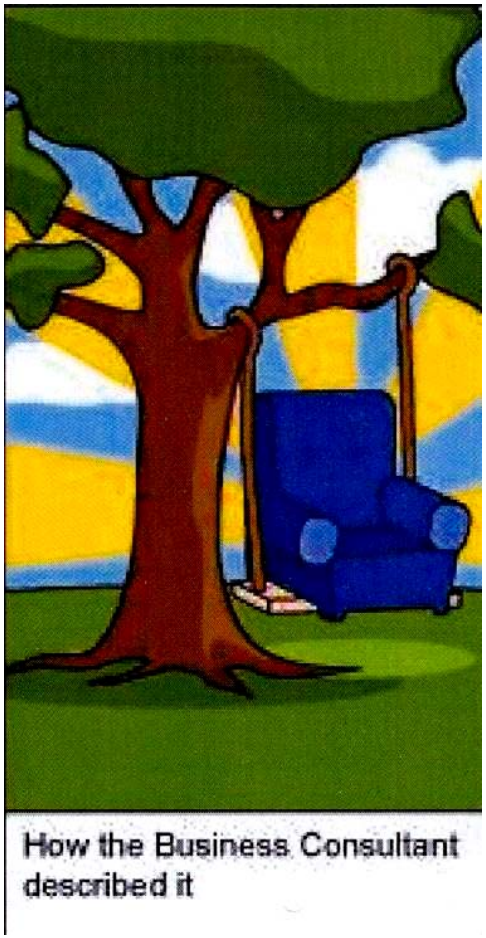
How the Analyst designed it

How the Analyst designed it



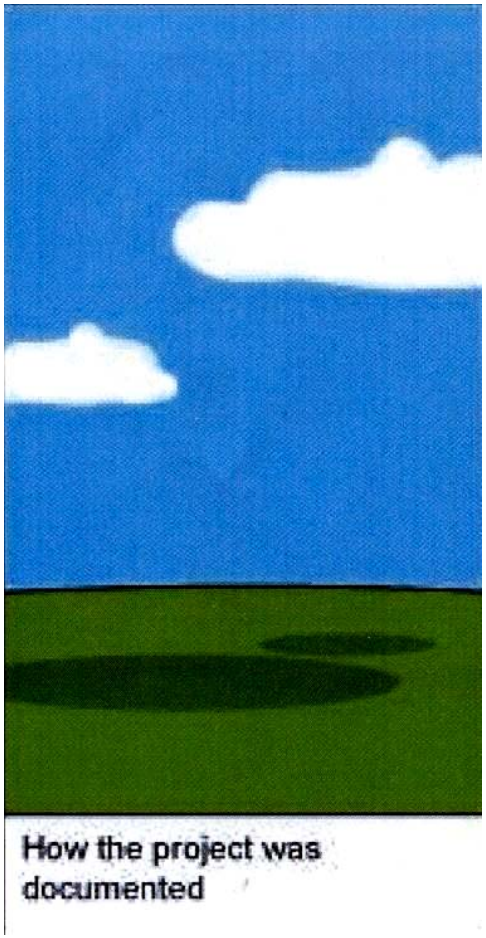
How the Programmer wrote it

How the Programmer Wrote it

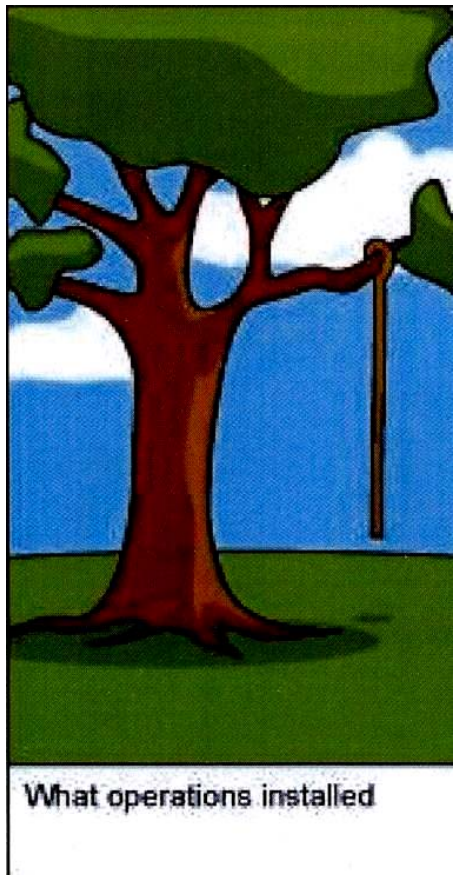


How the Business Consultant described it

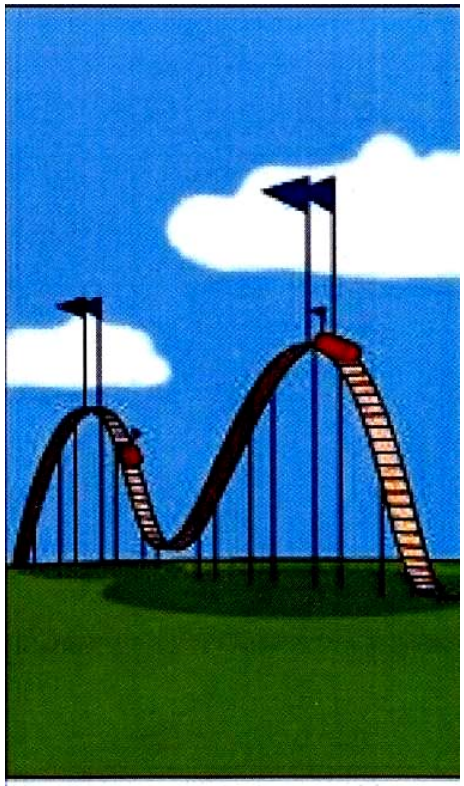
How the Business Consultant described it



How the project was documented

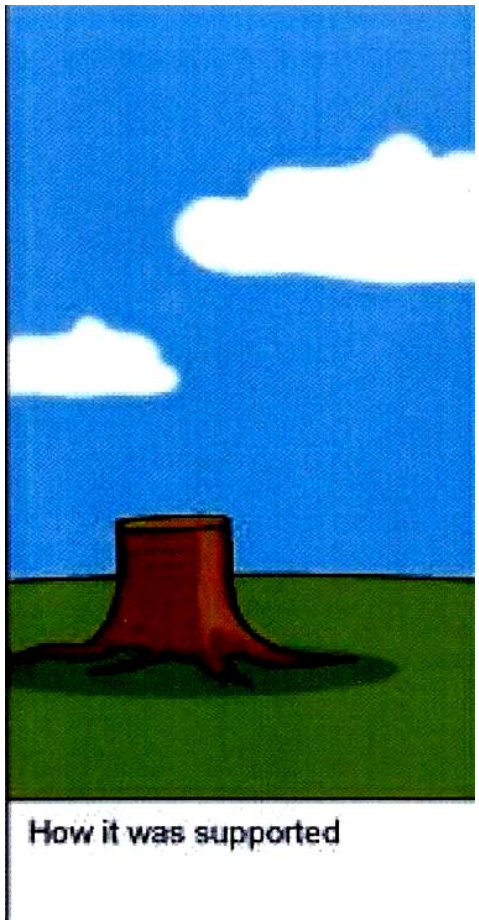


What operations installed

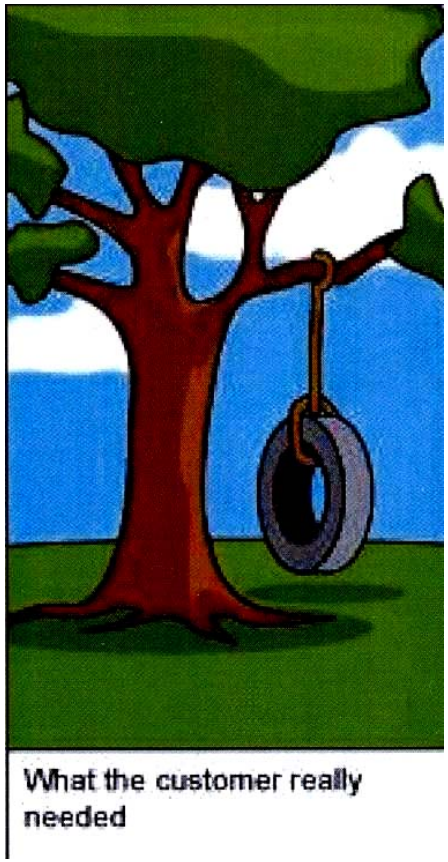


How the customer was billed

How the customer was billed



How it was supported



What the customer really needed

What are the costs of software engineering?

- Roughly 60% of costs are development costs, 40% are testing costs.
  For custom software, evolution costs often exceed development costs
- Costs vary depending on the type of system being developed and the requirements of system attributes such as performance and system reliability
- Distribution of costs depends on the development model that is used

What are the attributes of good software?

- The software should deliver the required functionality and performance to the user and should be maintainable, dependable and usable
- Maintainability
  - Software must evolve to meet changing needs
- Dependability
  - Software must be trustworthy
- Efficiency
  - Software should not make wasteful use of system resources
- Usability
  - Software must be usable by the users for which it was designed

The Activities of Software Engineering

Basic Activities of Software Engineering

- ***defining*** **the software development *process* to be used**
- ***managing*** **the development *project***
- ***describing*** **the intended software *product***
- ***designing*** **the *product***

Basic Activities of Software Engineering

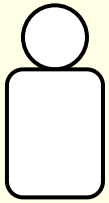
- ***implementing*** **the** ***product***

  i.e. programming it

- ***testing*** **the** ***parts*** **of the product**
- ***integrating*** **the parts and testing them as a whole**
- ***maintaining*** **the** ***product***

The Four “P’s” of Software Engineering

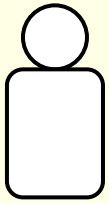
People



(by whom it is done)

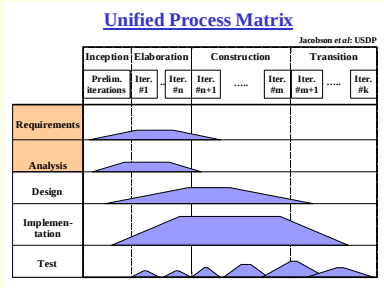
The Four “P’s” of Software Engineering

People



(by whom it is done)

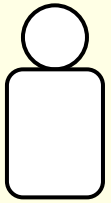
Process



(the manner in which it is done)

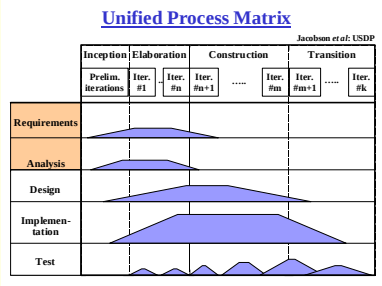
The Four “P’s” of Software Engineering

People



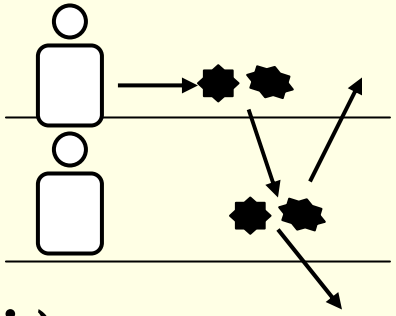
(by whom it is done)

Process



(the manner in which it is done)

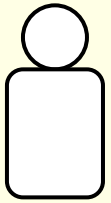
Project



(the doing of it)

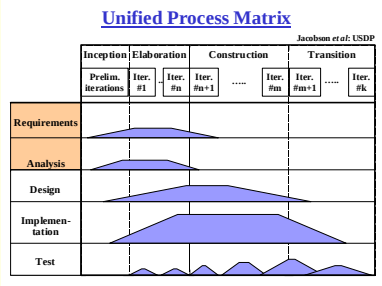
The Four “P’s” of Software Engineering

People



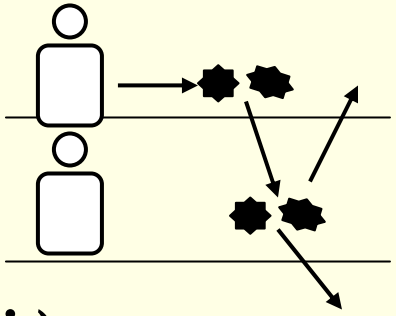
(by whom it is done)

Process



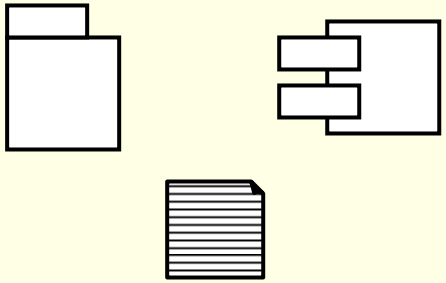
(the manner in which it is done)

Project



(the doing of it)

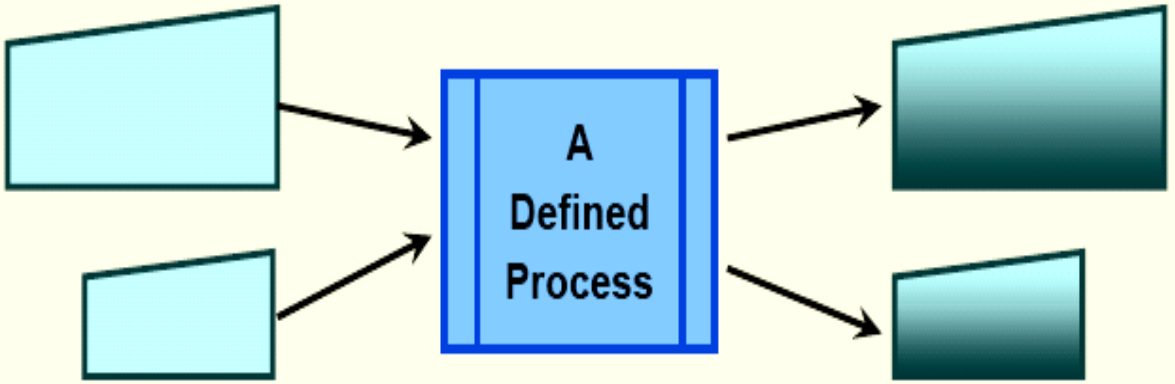
Product



(the application artifacts)

Process

A Defined Process



- Every task must be completely understood.
- When given a well-defined set of inputs, the same outputs are generated every time.

Tagasisideta süsteem

The Waterfall Model

Requirements analysis

Produces … specification (text)

Design

... diagrams & text

... code & comments

Implementation

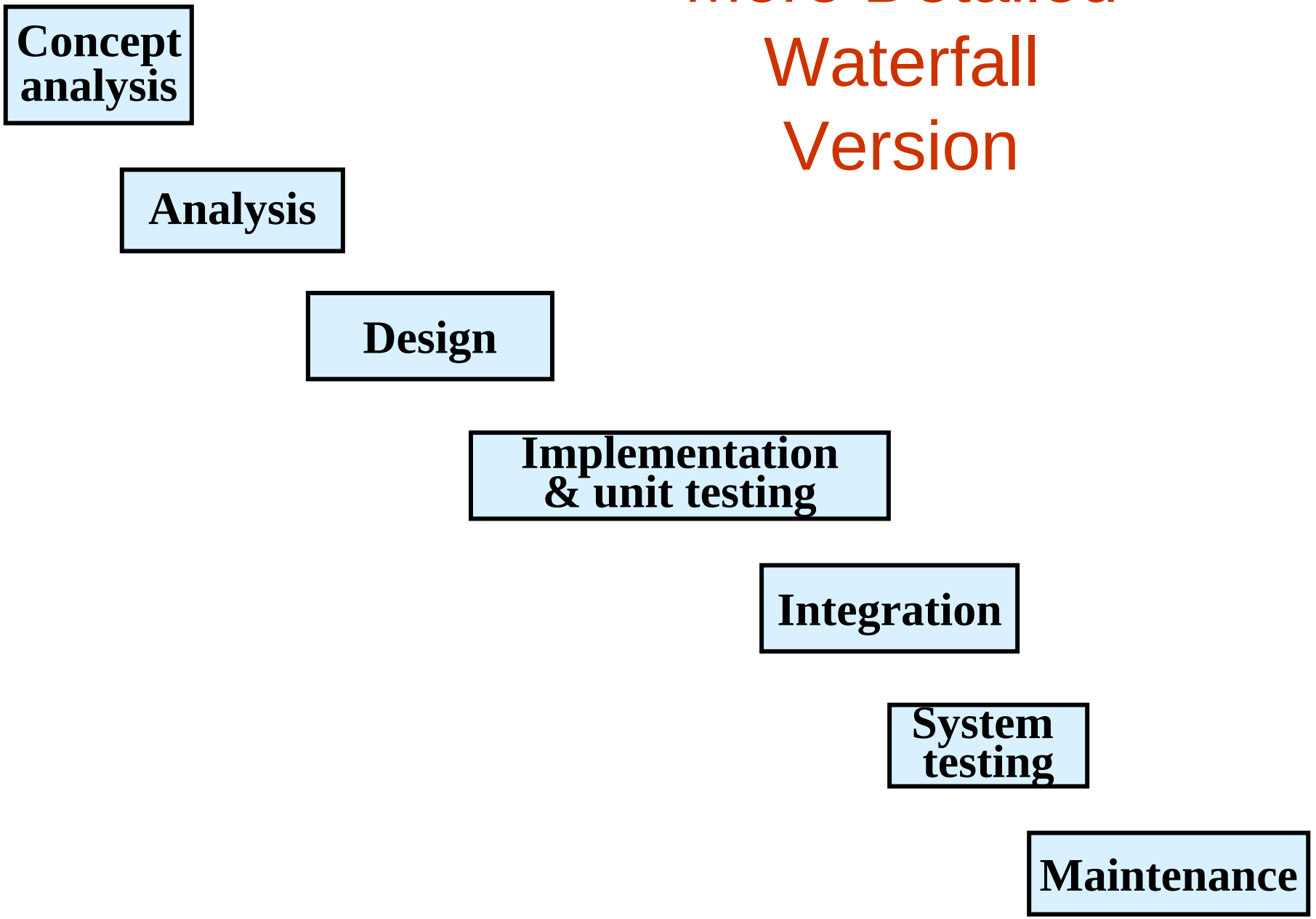
... entire code

Integration

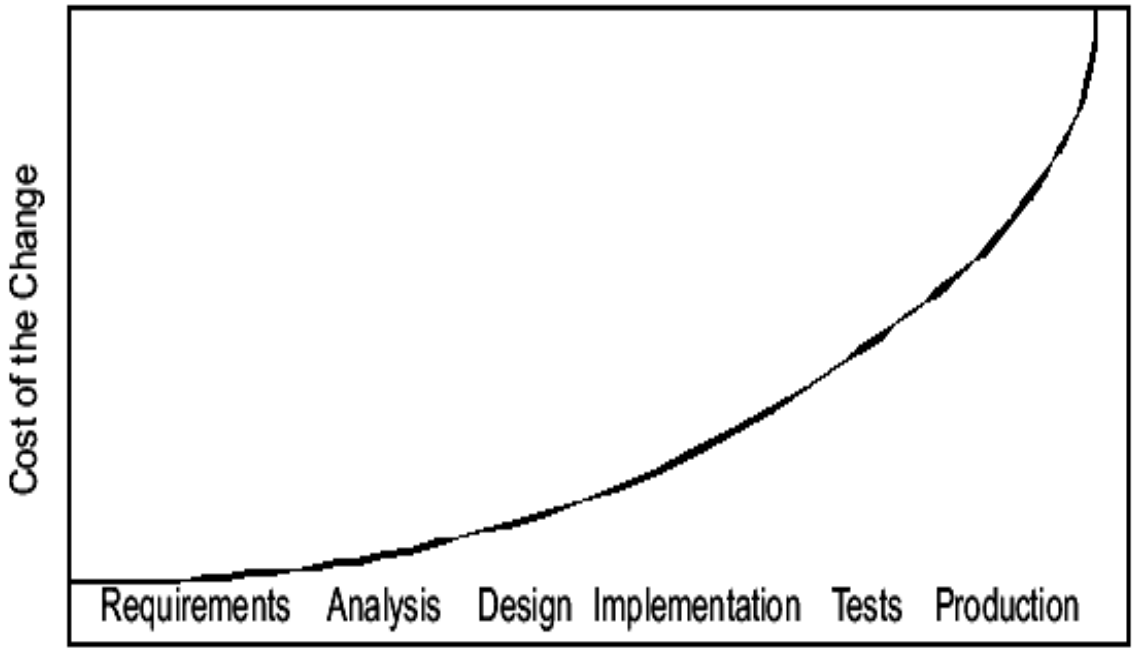
... test report, including defect descriptions

Test

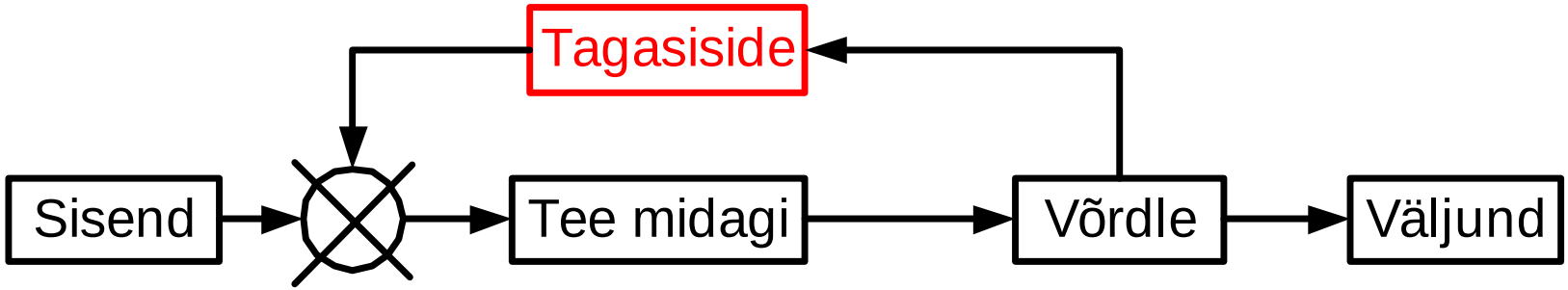
More Detailed Waterfall Version



Cost of change in "traditional" software engineering



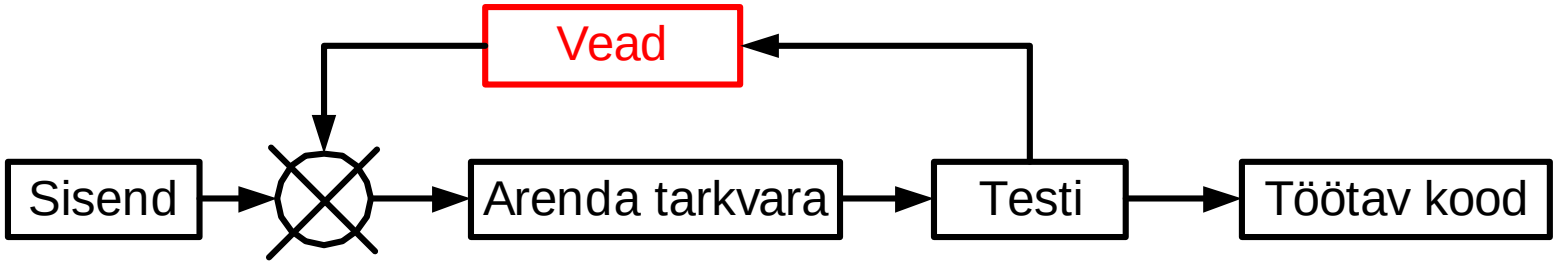
Tagasisidega süsteem



(Lihtne silmus)

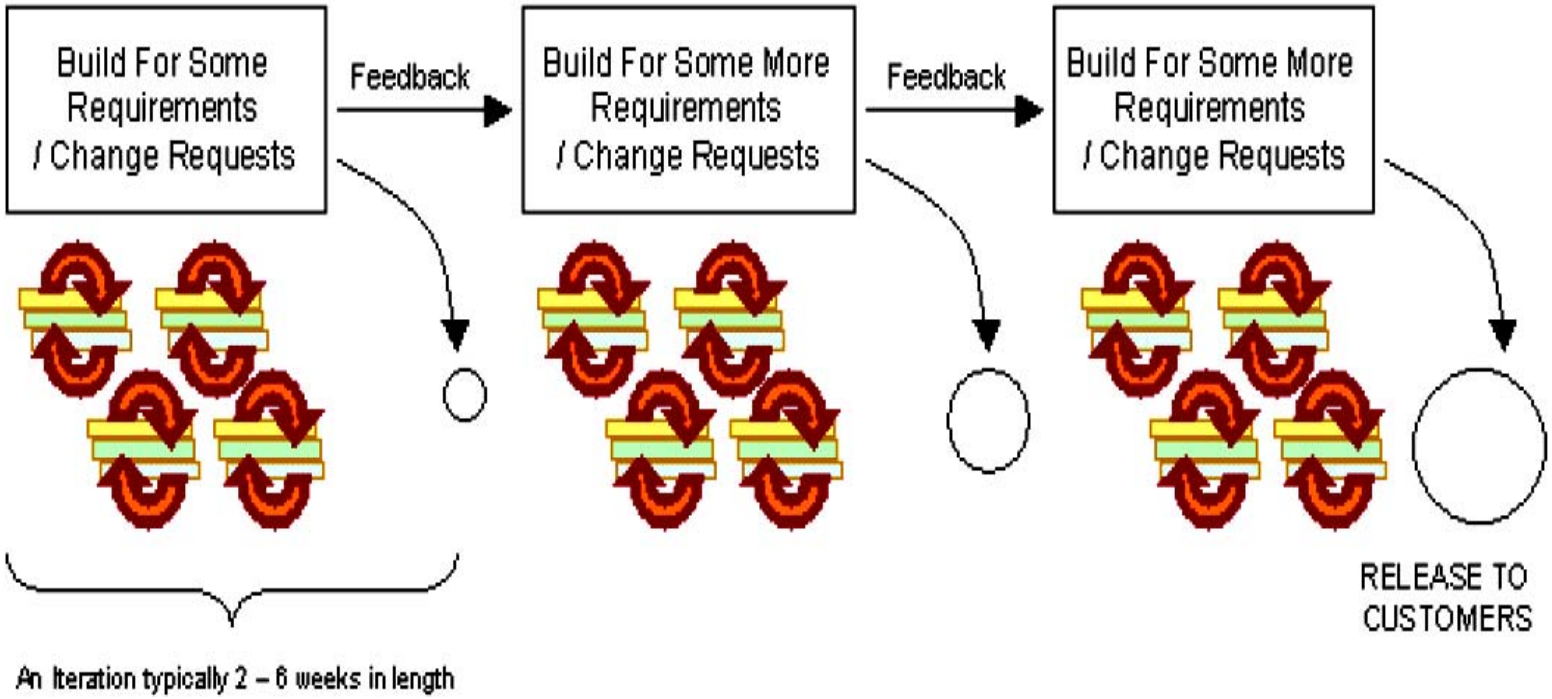
Tagasisidega süsteemidel on mitemeid eeliseid

Näide: Tarkvaraarendus kui süsteem

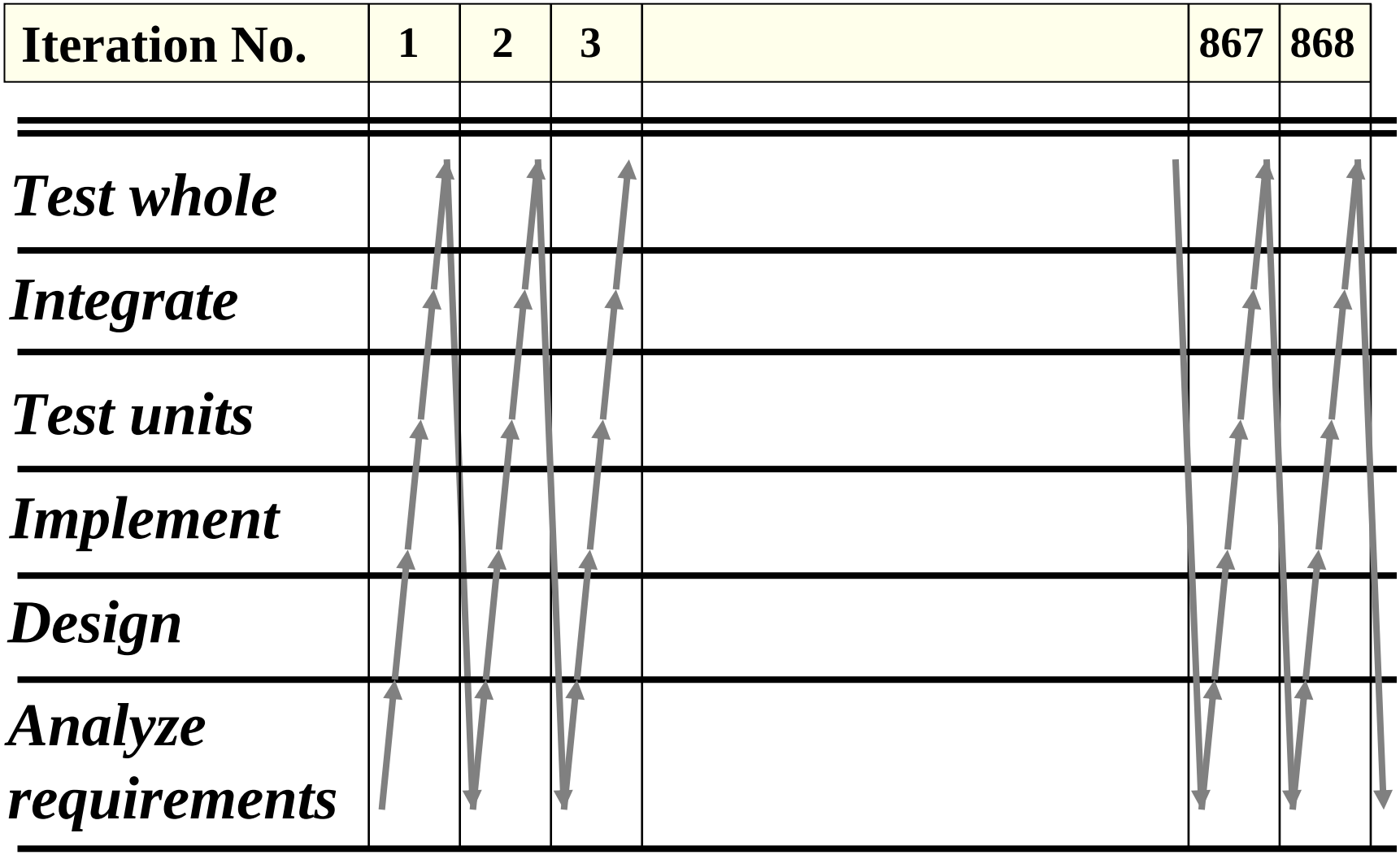


(Lihtne tarkvaraarenduse silmus)

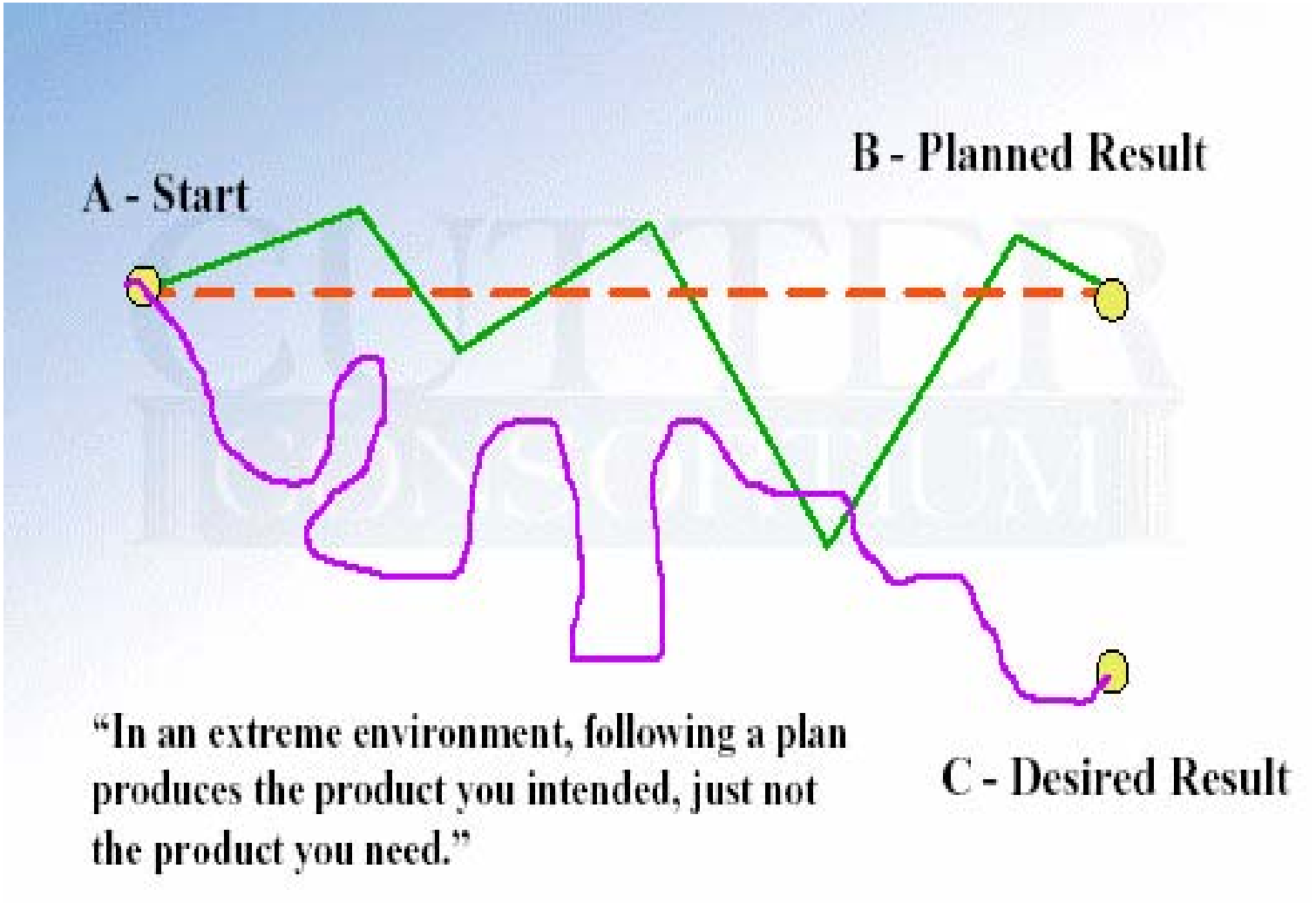
Iterative development



Incremental Development Sequence



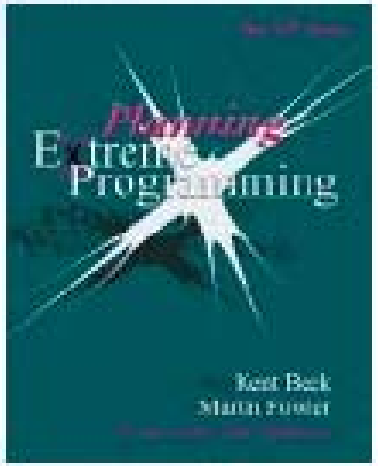
Iterative Development vs Waterfall

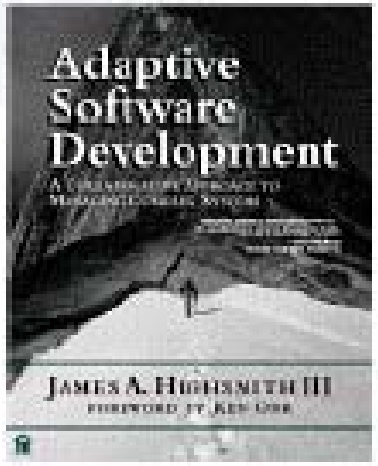


Agile Methodologies

Agile Methodologies

- Extreme Programming - Kent Beck +
- Crystal Methods - Alistair Cockburn
- Lean Development - Bob Charette
- SCRUM-K. Schwaber, J. Sutherland
- Adaptive Software Dev - Jim Highsmith





"I predict that Kent Beck and his XP movement will be as much a symbol of our times as Watts Humphry and the CMM were a symbol of the eighties and early nineties." - Tom DeMarco, Cutter Trends Report on Light Methodologies

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. We 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.

Signers: Arie van Bennekum (DSDM), Kent Beck (XP), Mike Beedle, Alistair Cockburn (Crystal), Ward Cunningham (XP), Martin Fowler (XP), James Grenning (XP), Jim Highsmith (ASD), Andy Hunt (Pragmatic Programming), Ron Jeffries (XP), Jon Kern (FDD), Brian Marick, Robert Martin (XP), Steve Mellor, Ken Schwaber, (SCRUM), Jeff Sutherland (SCRUM), PragmaticDave Thomas (Pragmatic Programming).

Simple Rules

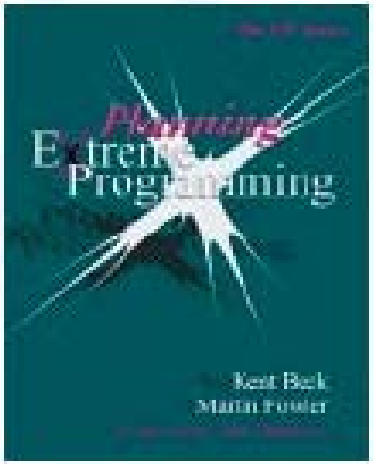
"Simple, clear purpose and principles give rise to complex, intelligent behavior."

"Complex rules and regulations give rise to simple, stupid behavior."

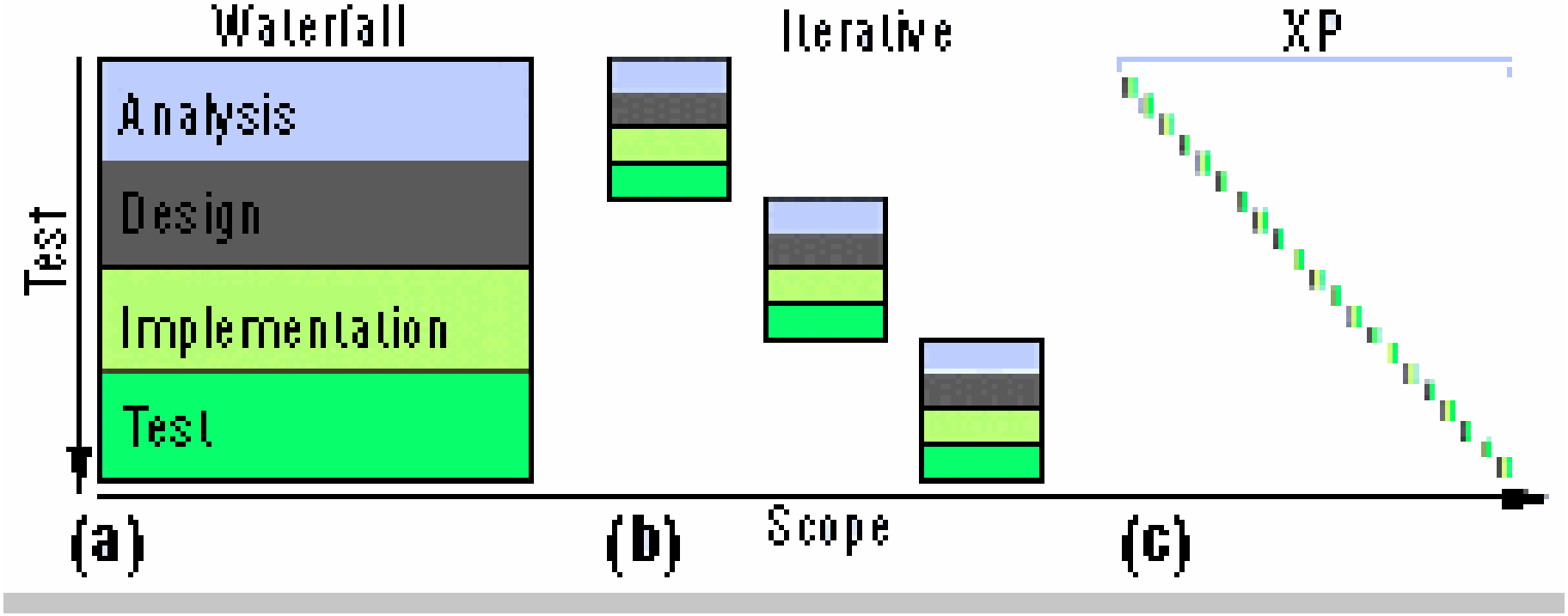
Agile Methodologies attempt to identify a few key practices (rules) and then let them evolve to meet specific problems through individual and group feedback.

Extreme Programming

- Best known of the Agile Methodologies
- Developed by 3 extremos--Kent Beck, Ward Cunningham, and Ron Jeffries
- Altered view of cost of change
- A system of practices
- Key new practices
  - refactoring
  - pair programming
  - organic (emergent design)
- References:
  - Extreme Programming: a series of books from Addison-Wesley
  - www.objectmentor.com (XP Training Workshops)
  - www.xprogramming.com (Ron Jeffries)



Kosemudel – iteratiivne arendusmudel – XP-arendusmudel



Lühikesed iteratsioonid

- Iteratsiooni kestus on tavaliselt kolm nädalat, igal iteratsioonil antakse kliendile töötava süsteemi (nudi-) versioon.
- Nii saab klient otsekohe loodava süsteemiga tööd alustada ja teha uute *lugude* kaudu ettepanekuid märgatud puuduste kõrvaldamiseks ja süsteemi täiendamiseks järgnevates iteratsioonides.

Lihtne kavand

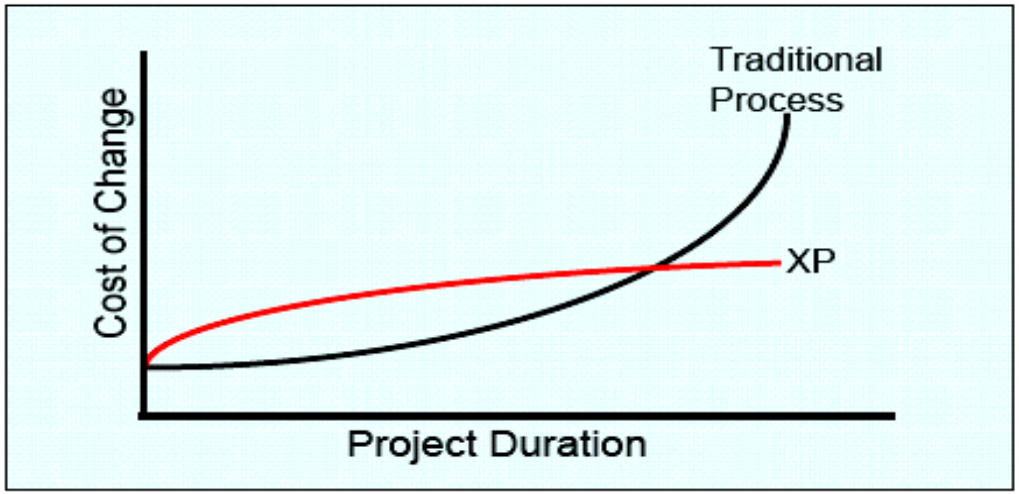
- Lihtne kood tagab lihtsa kavandi (ja *vice versa*).
- Koodi ülevaatustel avastatud lihtsusprintsiibist kõrvalekaldumised likvideeritakse koheselt.

Testimine

- Klient kirjutab teste oma *lugude* verifitseerimiseks.
- Programmeerijad kirjutavad teste, et avastada vigu kodeerimisel.
- Testid kirjutatakse enne kodeerimist.

The Cost of change

- "The error [is] typically 100 times more expensive to correct in the maintenance phase than in the requirements phase."
  - *Software Engineering Economics*, Barry Boehm, 1981, p. 40.



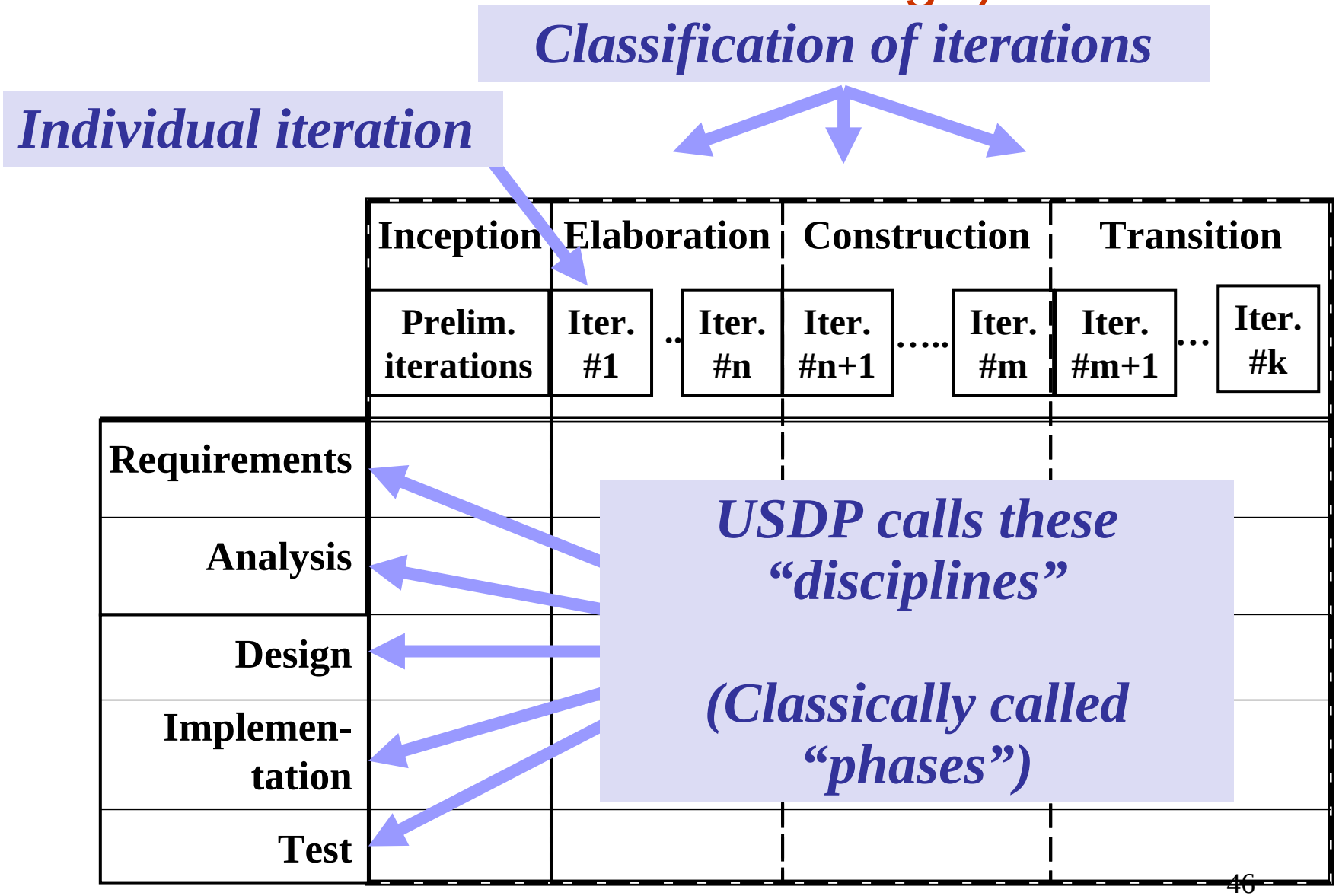
Pair Programming

Rational Unified Process (RUP)

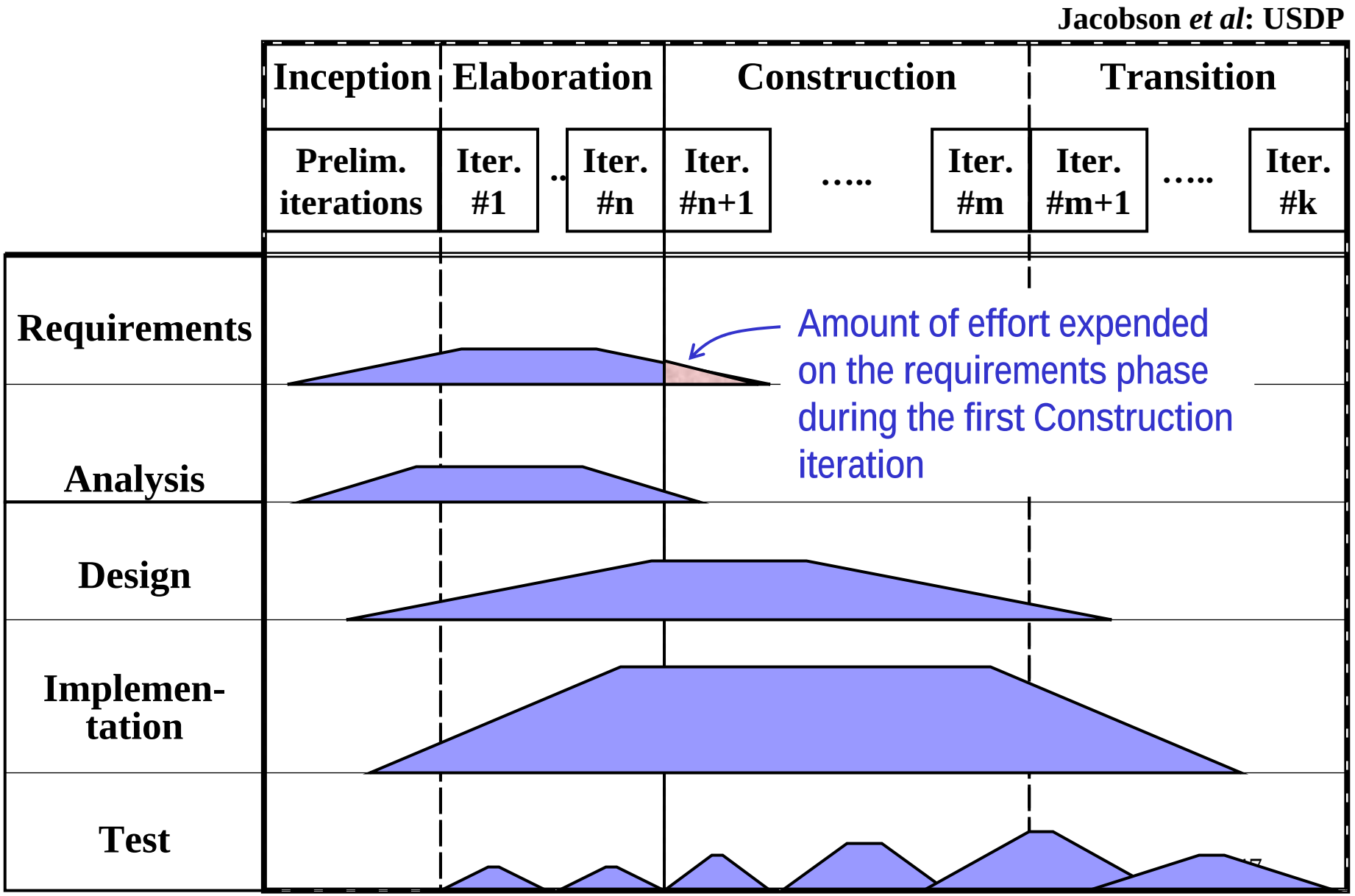
The Unified Software Development Process: Classification of Iterations

- ***Inception iterations*: preliminary interaction with stakeholders**
  - **primarily customer**
  - **users**
  - **financial backers etc.**
- ***Elaboration iterations* : finalization of what's wanted and needed; set architecture baseline**
- ***Construction iterations* : results in initial operational capability**
- ***Transition iterations* : completes product release**

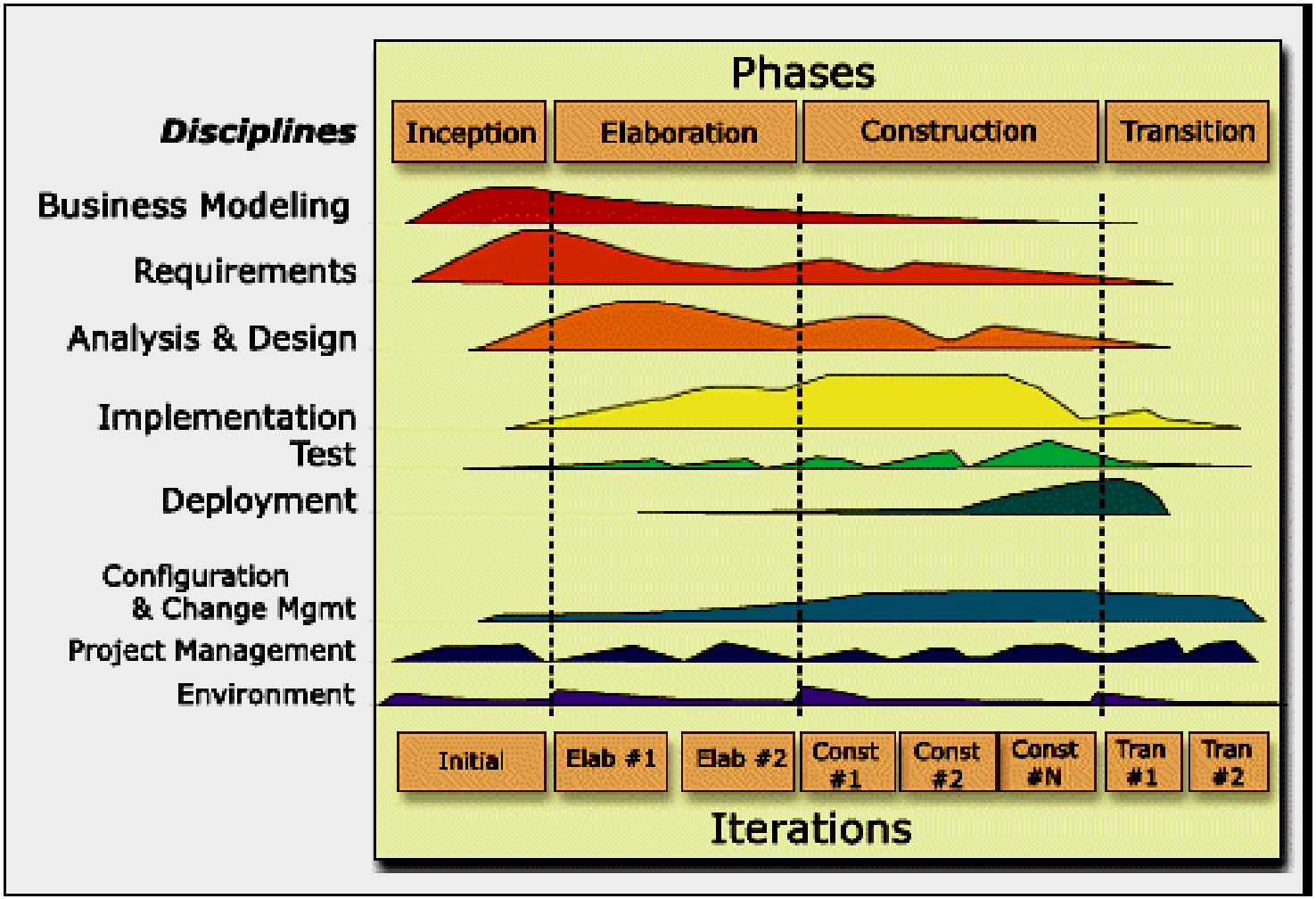
USDP vs. Standard Terminology ½ (Booch, Jacobson & Rumbaugh)



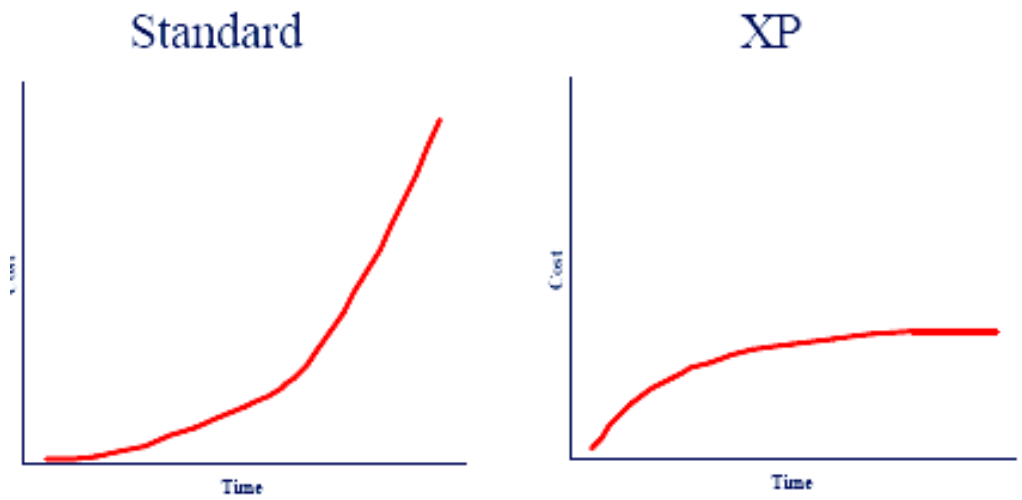
Unified Process Matrix

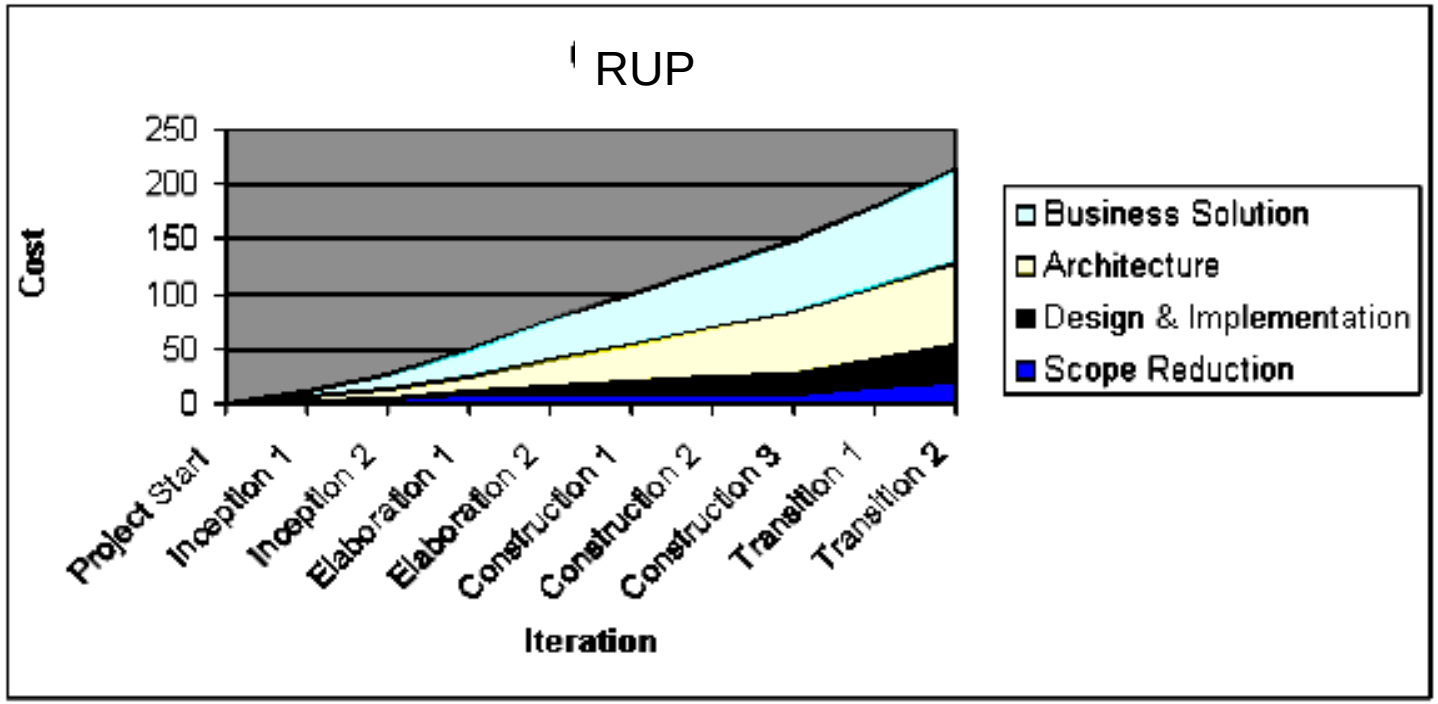


RUP—A Well-Defined Software Engineering Process

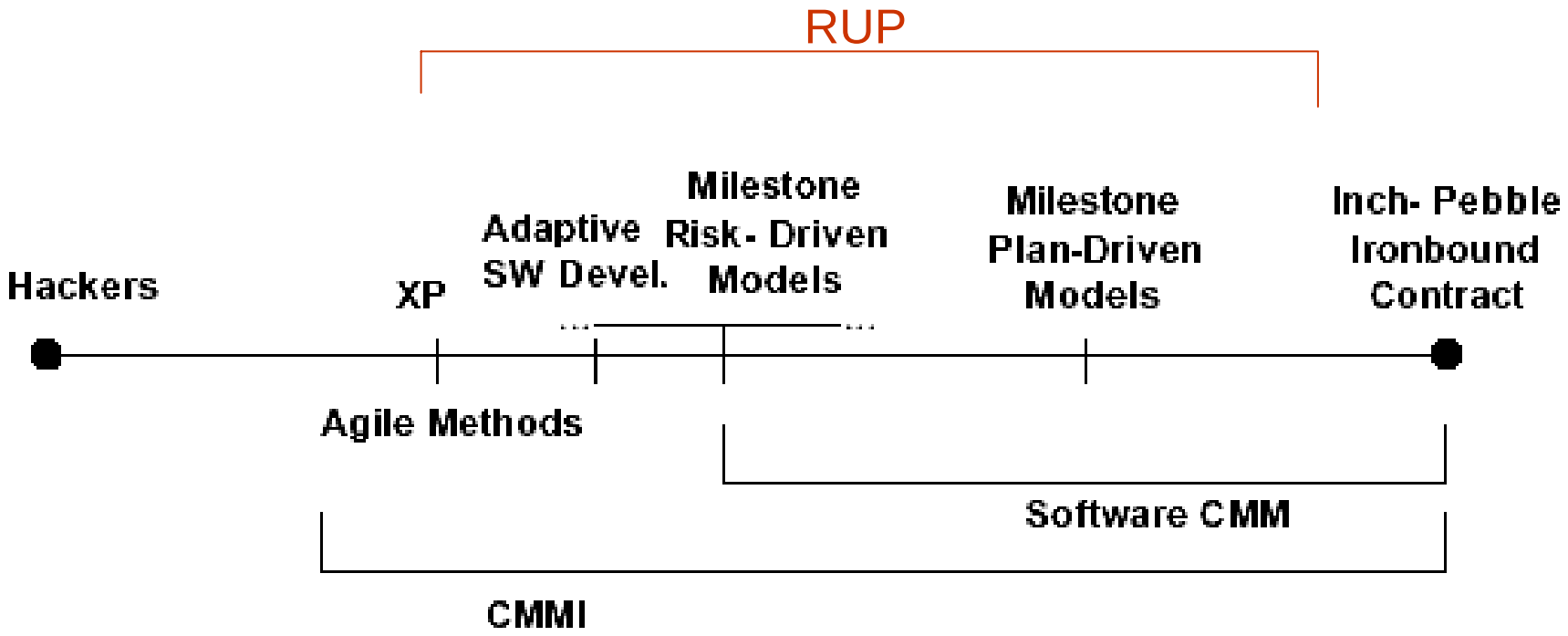


Cost of change

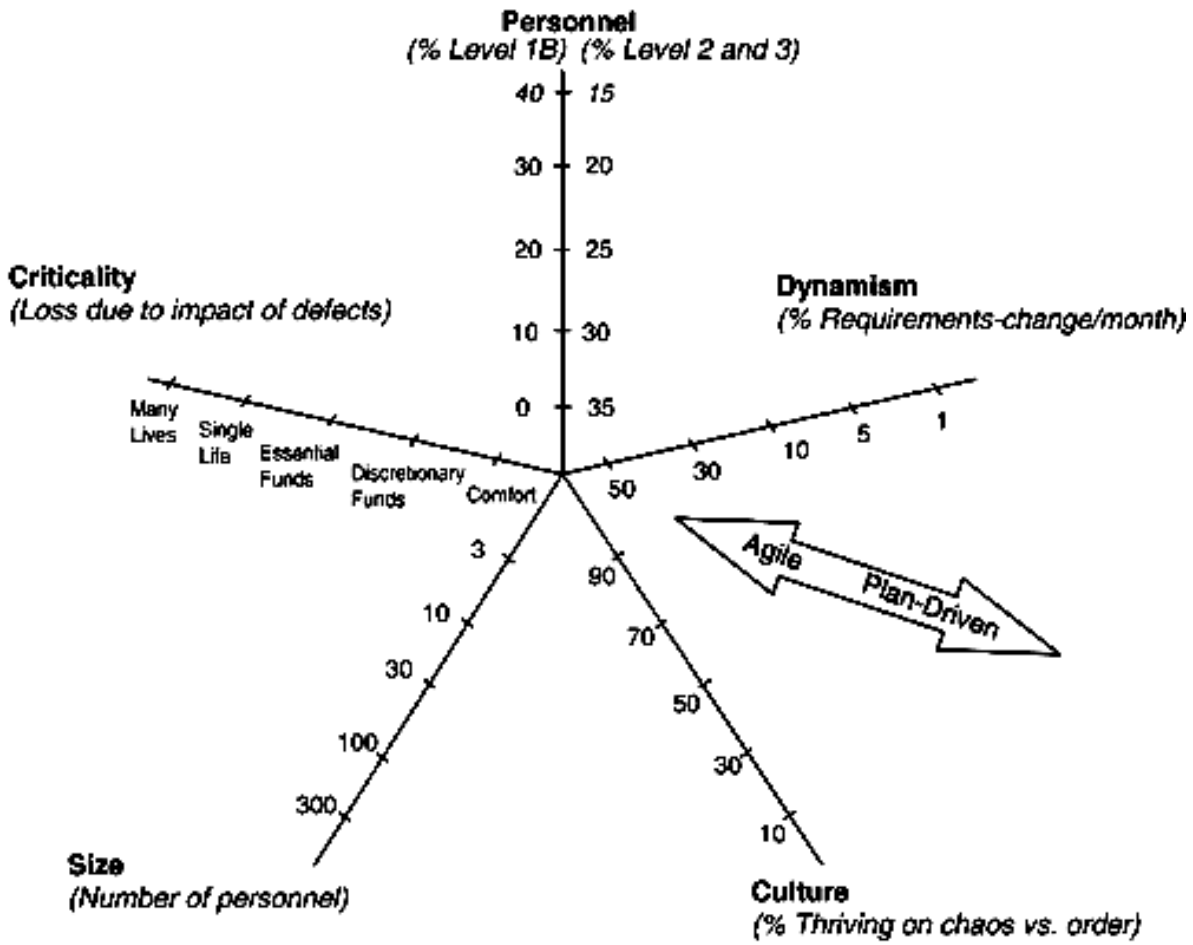




Methods' scopes



Dimensions Affecting Method Selection

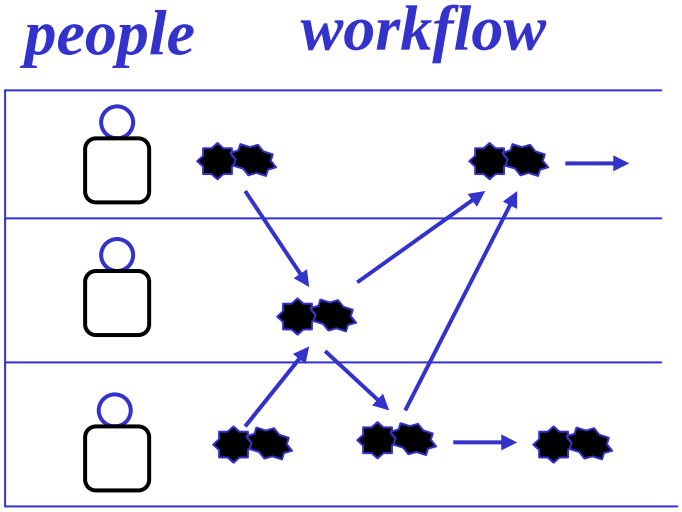


Project

Project

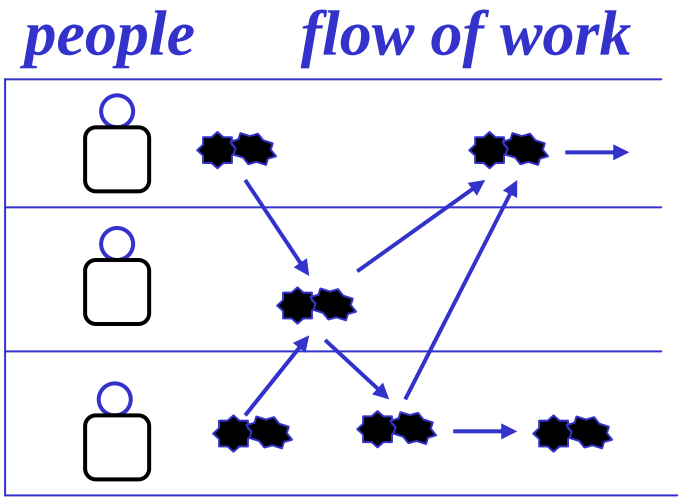
Set of activities carried out to produce an application

- ***OO*: very useful paradigm**



Project

Set of activities carried out to produce an application



- ***Object Orientation*: very useful paradigm**
- ***Unified Modeling Language*: design notation**
- ***Legacy systems:* common starting point**
  - **enhancement or usage of existing system**

Agility - agiilsus

- *Agility is the ability to both create and respond to change in order to profit in a turbulent business environment.*
- *Agility is the ability to balance flexibility and stability*
- Agiilsus:
  - *planeerimisele-plaani täitmisele* vastandab *visiooni loomise – uuringu*
  - *ettemääratusele* vastandab *adaptiivsuse*

Simple rules = simplicity

- **Rule No 1.** *Use your good judgment in all situations.*

There will be no additional rules

Tasakaal kaose ja korra vahel

- Edukas (projekti)juhtimine saavutab tasakaalu kaose ja korra vahel, seega on meil vaja pisut enam reegleid, et ka teatavat korda tagada.
- Kuid iga reegli lisamine võib pärssida tiimi innovatiivsust ja võimet võtta vastutust ...

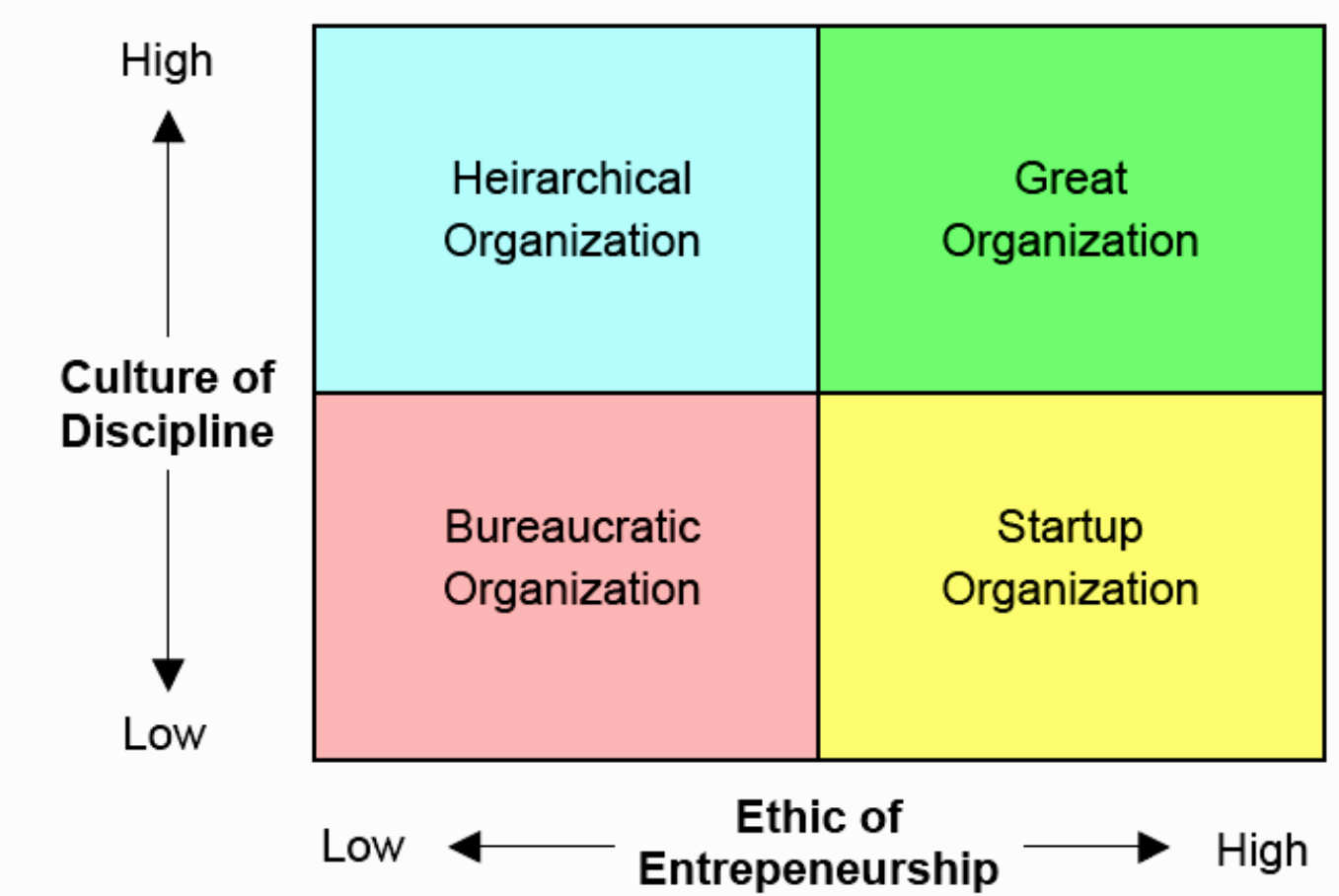
Klassikaline juhtimisteooria eeldab

- Hierarhilised organisatsiooni struktuurid on vahendiks korra loomisel
- Mida enam kontrolli, seda enam korda
- Töötajad on „vahetatavad osad“ organisatsiooni „masinas“
- Probleemid lahendatakse „jaga ja valitse“ printsiibil
- Projekti täitmise kulg ja riskid on piisavalt prognoositavad ja seega hallatavad põhjaliku eelneva plaanimisega

Agiilprojektid

- Agiilne projektijuhtimine sobib nendes valdkondades ja nendes projektides (agiilprojektides), mille lähtetingimused on ähmased (*fuzzy*), mille eesmärkide saavutamiseks on vajalik innovatsioon, millegi avastamine, st uurimuslik tarkvaraarendus.

Matrix of Creative Discipline



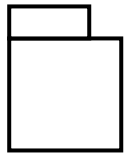
Product

Product -- the application and associated artifacts, including:

- ***Requirements***
  - **explain what product is meant to be**
- ***Software architecture***
  - **explains product's design principles**
- ***Detailed design***
  - **explains how product is meant to made**

Artifacts

Software Requirements Specification



Design model

Product -- the application, and associated artifacts, including:

- ***Requirements***

  explain what product is meant to be

- ***Software architecture***
- ***Detailed design***

  use the language of Design Patterns

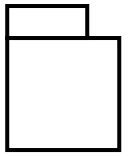
- ***Implementation***

  emphasize standards
  employ selected formal methods.

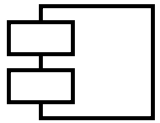
- ***Test artifacts***

Artifacts

Software Requirements Specification



Design model



Source & object code

Test procedures; test cases

Quality

Application must satisfy predetermined standards.

Methods to attain quality goals:

Quality

- ***Inspection***
  - **team-oriented process for ensuring quality**
  - **applied to all stages of the process.**

Application must satisfy predetermined quality level.

Methods to attain quality level: Quality

- ***Inspection***
  - team-oriented process for ensuring quality
  - applied to all stages of the process.
- ***Formal methods***
  - mathematical techniques to convince ourselves and peers that our programs do what they are meant to do
  - applied selectively
- ***Testing***
  - at the unit (component) level
  - at the whole application level
- ***Project control techniques***
  - predict costs and schedule
  - control artifacts (versions, scope etc.)

The Capability Maturity Model (CMM)

1. Initial

Process:

undefined,

ad hoc

Result:

outcome depends on individuals

Lacking: **any reasonable process**

1. **INITIAL** **Process undefined, ad hoc, depends on individuals**

2. Repeatable

02.02

Process

tracks documents, cost, schedule, functionality (after fact)

Result

repeatable only on similar projects

Lacking: **complete process**

2. REPEATABLE Basic project management to track cost & schedule, repeatable on similar projects

3. Defined

Process

documented,

standardized,

tailorable

Result

consistency

***Lacking*: predictable outcomes**

3. DEFINED **Consistent: Documented, standardized, tailorable**

4. Managed

Process

detailed measurement; control

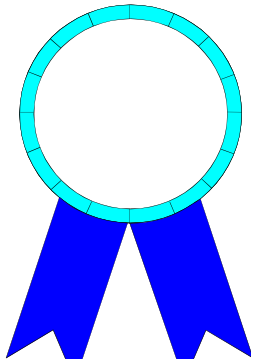
Result

process and products with quantified quality predictability

Lacking **mechanism for process improvement**

4. MANAGED Predictable: process & products measured

5 Optimized



Process

Continual process improvement

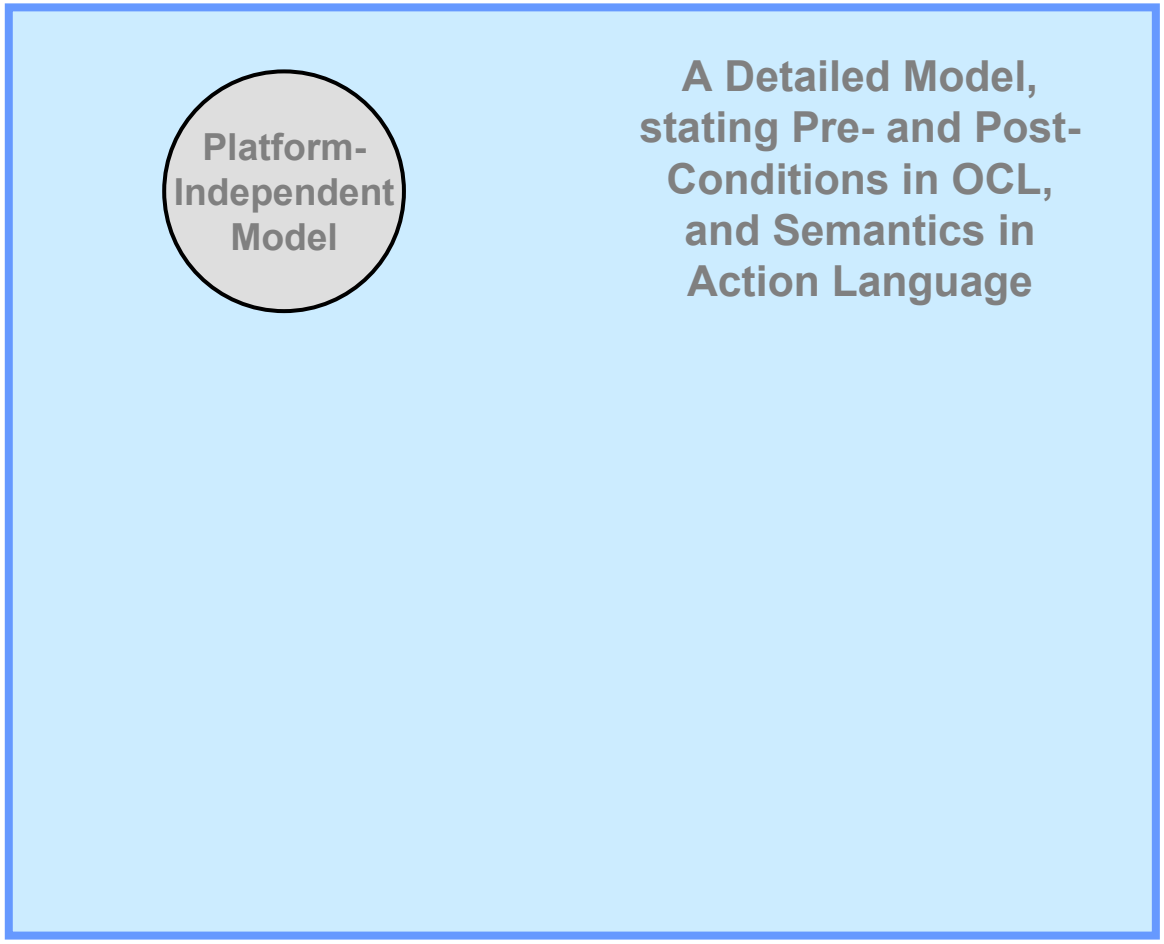
through quantitative feedback;

Extensible scope

Innovative ideas and technologies

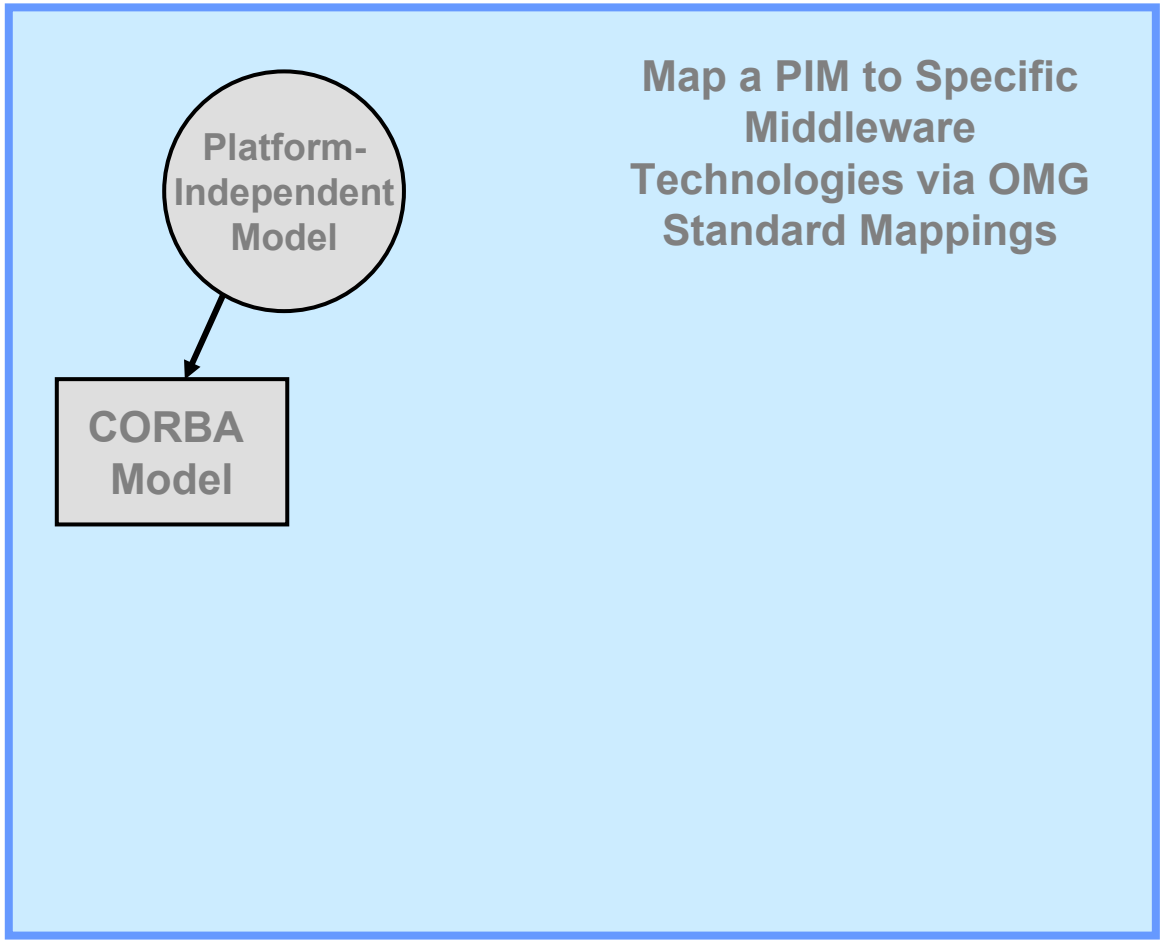
Model Driven Architecutre (MDA)

Building an MDA Application



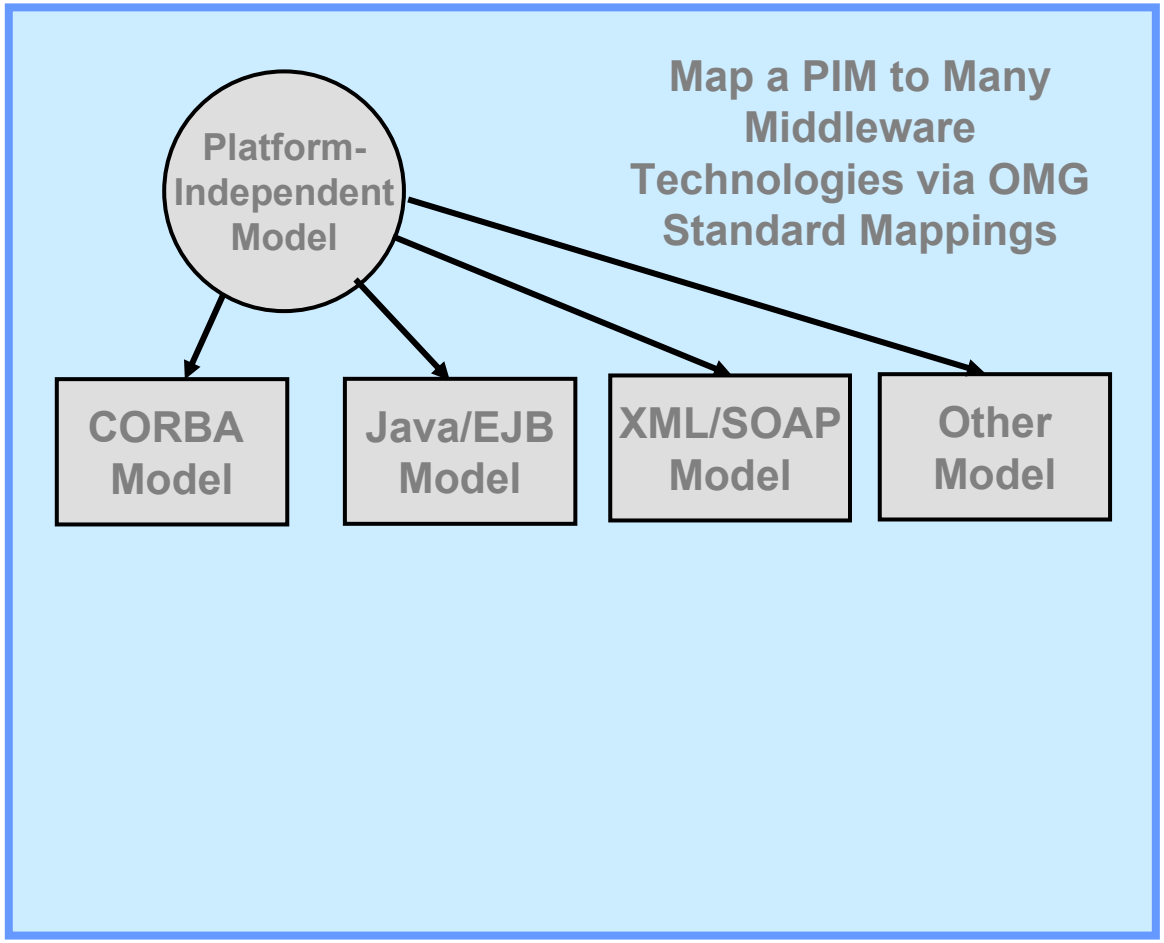
Start with a *Platform-Independent Model* (PIM) representing business functionality and behavior, undistorted by technology details.

Generating Platform-Specific Model



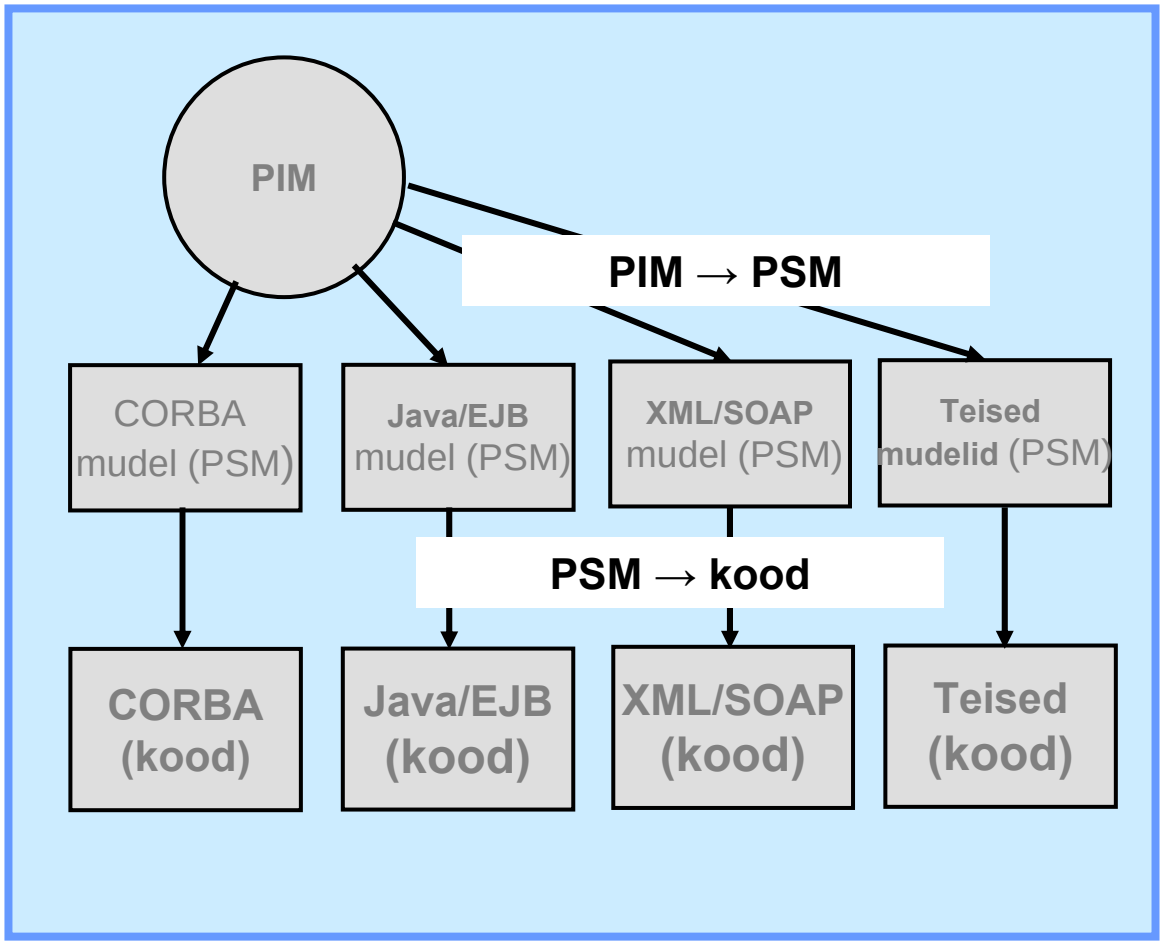
MDA tool applies a standard mapping to generate *Platform-Specific Model* (PSM) from the PIM. Code is partially automatic, partially hand-written.

Mapping to Multiple Deployment Technologies



MDA tool applies an standard mapping to generate *Platform-Specific Model* (PSM) from the PIM. Code is partially automatic, partially hand-written.

Generating Implementations



MDA Tool generates all or most of the implementation code for deployment technology selected by the developer.