*L3 Mention Informatique*
*Parcours Informatique et MIAGE*

# Génie Logiciel Avancé - Advanced Software Engineering

## An Introduction

Burkhart Wolff
wolff@lri.fr

## What is Software Engineering ?

## What is Software Engineering ?

## What is Software Engineering ?

- Methods, techniques and tools for

## What is Software Engineering ?

- Methods, techniques and tools for
  - design: requirement analysis, models, specifications

## What is Software Engineering ?

- Methods, techniques and tools for
  - design: requirement analysis, models, specifications
  - development: programmation, integration

## What is Software Engineering ?

- Methods, techniques and tools for
  - design: requirement analysis, models, specifications
  - development: programmation, integration
  - validation: prototypes, testing

## What is Software Engineering ?

- Methods, techniques and tools for
  - design: requirement analysis, models, specifications
  - development: programmation, integration
  - validation: prototypes, testing
  - verification: formal proof of required properties

## What is Software Engineering ?

- Methods, techniques and tools for
  - design: requirement analysis, models, specifications
  - development: programmation, integration
  - validation: prototypes, testing
  - verification: formal proof of required properties
  - maintenance: reusability, improvements

## What is Software Engineering ?

- Methods, techniques and tools for
  - design: requirement analysis, models, specifications
  - development: programmation, integration
  - validation: prototypes, testing
  - verification: formal proof of required properties
  - maintenance: reusability, improvements

## What is Software Engineering ?

- Methods, techniques and tools for
  - design: requirement analysis, models, specifications
  - development: programmation, integration
  - validation: prototypes, testing
  - verification: formal proof of required properties
  - maintenance: reusability, improvements

- A slightly longer answer:

## What is Software Engineering ?

- Methods, techniques and tools for
  - design: requirement analysis, models, specifications
  - development: programmation, integration
  - validation: prototypes, testing
  - verification: formal proof of required properties
  - maintenance: reusability, improvements

- A slightly longer answer:

## What is Software Engineering ?

## What is Software Engineering ?

## What is Software Engineering ?

- … slightly longer answer:

## What is Software Engineering ?

- … slightly longer answer:

  The discipline of software engineering was created to address poor quality of software, get projects exceeding time and budget under control, and ensure that software is built systematically, rigorously, measurably, on time, on budget, and within specification. [Wikipedia [en]]

## What is Software Engineering ?

- ... slightly longer answer:

  The discipline of software engineering was created to address poor quality of software, get projects exceeding time and budget under control, and ensure that software is built systematically, rigorously, measurably, on time, on budget, and within specification. [Wikipedia [en]]

- Or much shorter:

## What is Software Engineering ?

- ... slightly longer answer:

  The discipline of software engineering was created to address poor quality of software, get projects exceeding time and budget under control, and ensure that software is built systematically, rigorously, measurably, on time, on budget, and within specification. [Wikipedia [en]]

- Or much shorter:

  SE addresses the problems of

  « Development in the Large »

  ... so for teams with 100 or 1000 of developers, and budgets of sometimes billions of dollars.

## What is Software Engineering ?

## What is Software Engineering ?

- … lets consider this more closely:

  The discipline of software engineering was created to address poor quality of software, get projects exceeding time and budget under control, and ensure that software is built systematically, rigorously, measurably, on time, on budget, and within specification. [Wikipedia [en]]

## What is Software Engineering ?

- … lets consider this more closely:

  The discipline of software engineering was created to address poor quality of software, get projects exceeding time and budget under control, and ensure that software is built systematically, rigorously, measurably, on time, on budget, and within specification. [Wikipedia [en]]

## What is Software Engineering ?

## What is Software Engineering ?

- poor quality
- projects exceeding
  time and budget
- software is built systematically,
- ... within specification.

## What is Software Engineering ?

Covered in this course:

- poor quality
- projects exceeding
  time and budget
- software is built systematically,
- ... within specification.

## What is Software Engineering ?

Covered in this course:

- poor quality ✔✔
- projects exceeding
  time and budget
- software is built systematically,
- ... within specification.

## What is Software Engineering ?

| | Covered in this course: |
|---|---|
| ❑ poor quality | ✔✔ |
| ❑ projects exceeding time and budget | - |
| ❑ software is built systematically, | |
| ❑ ... within specification. | |

## What is Software Engineering ?

Covered in this course:

- poor quality ✔✔
- projects exceeding time and budget -
- software is built systematically,
- ... within specification.

for something like *constructive cost models* (COCOMO) you need experience with a concrete process !

## What is Software Engineering ?

Covered in this course:

- poor quality ✔✔
- projects exceeding time and budget -
- software is built systematically, ✔
- ... within specification.

for something like *constructive cost models* (COCOMO) you need experience with a concrete process !

# What is Software Engineering ?

Covered in this course:

- poor quality ✔✔
- projects exceeding time and budget -
- software is built systematically, ✔
- ... within specification. ✔✔

for something like *constructive cost models* (COCOMO) you need experience with a concrete process !

## The Problem for Quality: Size

## The Problem for Quality: Size

- Some (anecdotal) empirical data based on the common criteria “lines of code” (LOC)

## The Problem for Quality: Size

- Some (anecdotal) empirical data based on the common criteria "lines of code" (LOC)

| Software | Software Type | Year (approx) | LOC | |
|---|---|---|---|---|
| Space Shuttle | Embedded + Services | 1980 | 50M | |
| Windows 90 | OS | 1990 | 10M | |
| Peugeot 607 | Embedded | 2000 | 2M | |
| Win2000 | OS | 2000 | 30M | |
| Hyper V | V-OS (Azure core) | 2008 | 50K | |
| X Window | Unix Windowing Sys. | 2008 | 1.8M | |
| Azure | Virtualising Services | 2009 | 110M | |
| Mozilla Firefox | Browser (Application) | 2020 | 23M | * |
| | | | | |

* https://www.openhub.net/p/firefox/analyses/latest/languages_summary

## The Problem for Quality: Size + Aging

## The Problem for Quality: Size + Aging

- Some (anecdotal) empirical data on one product-line for a core-product of a Business SW Company (SAP)

# The Problem for Quality: Size + Aging

- Some (anecdotal) empirical data on one product-line for a core-product of a Business SW Company (SAP)

| Software | Software Type | Year (approx) | LOC |
|---|---|---|---|
| Version 1 | Business Application Suite | 1973 | ??? |
| Version 5.1 | Business Application Suite | 2004 | <60M |
| Version 5.2 | Business Application Suite | 2000 | 90M |
| Version 6.0 | Business Application Suite | 2000 | 100M |
| Version 7.0 | Business Application Suite | 2008 | 105M |
| Version 7.1 | Business Application Suite | 2008 | 118 |
| | | | |
| | | | |
| | | | |

## The Problem for Quality: Changing “Company Cultures”

https://www.transparencymarketresearch.com/automotive-software-market.html

## The Problem for Quality: Changing "Company Cultures"

https://www.transparencymarketresearch.com/automotive-software-market.html

## The Problem for Quality: Changing "Company Cultures"

- Software is invasive. In many traditional "engineering" companies, most of the money is made by software.

https://www.transparencymarketresearch.com/automotive-software-market.html

## The Problem for Quality: Changing "Company Cultures"

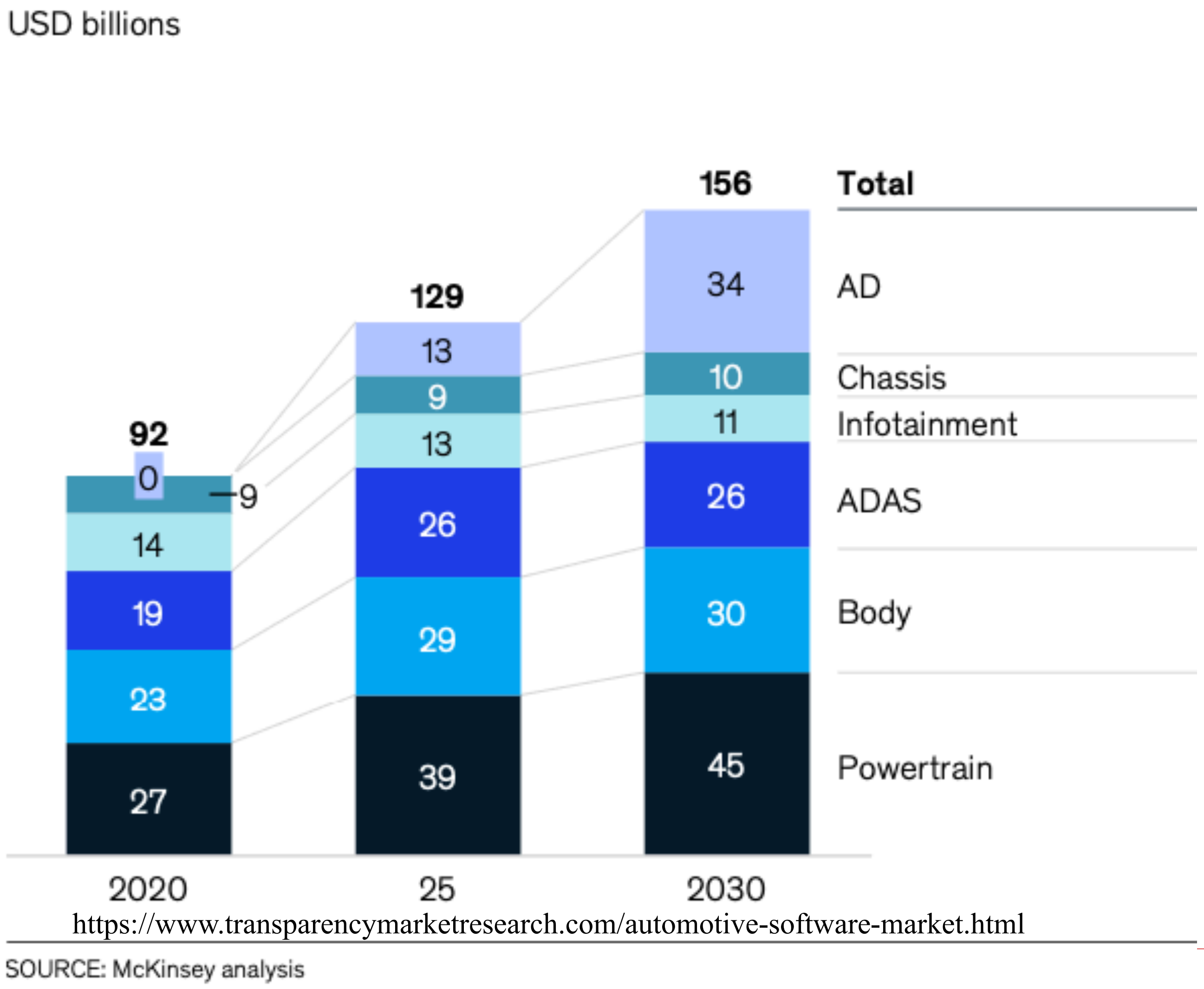
- Software is invasive. In many traditional "engineering" companies, most of the money is made by software.
- Example: Automotive.

https://www.transparencymarketresearch.com/automotive-software-market.html

## The Problem for Quality: Changing “Company Cultures”

https://www.transparencymarketresearch.com/automotive-software-market.html

# The Problem for Quality: Changing "Company Cultures"



https://www.transparencymarketresearch.com/automotive-software-market.html

## The Problem for Quality: Changing "Company Cultures"

## The Problem for Quality: Changing "Company Cultures"

## The Problem for Quality: Changing “Company Cultures”

- But become traditional “engineers” automatically “software engineers” ???

## The Problem for Software-Quality

## The Problem for Software-Quality

## The Problem for Software-Quality

- A Very General Rule of Thumb:

## The Problem for Software-Quality

- A Very General Rule of Thumb:
  - Programming is not enough ! Overall,
    It is not even the most important cost-factor !!

## The Problem for Software-Quality

- A Very General Rule of Thumb:
  - Programming is not enough ! Overall,
    It is not even the most important cost-factor !!
  - A global estimate of project activities:

    Percentage of «Coding» ? 15 - 20 %
    Proportion of Validation et Verification ? ~20%
    All others : (Analysis,Design, Certification,
    Maintenance, Management). 60 %

## The Problem for Software-Quality

- A Very General Rule of Thumb:
  - Programming is not enough ! Overall,
    It is not even the most important cost-factor !!
  - A global estimate of project activities:

| | |
|---|---|
| Percentage of «Coding» ? | 15 - 20 % |
| Proportion of Validation et Verification ? | ~20% |
| All others : (Analysis,Design, Certification, Maintenance, Management). | 60 % |

## The Problem for Software-Quality

- A Very General Rule of Thumb:
  - Programming is not enough ! Overall,
    It is not even the most important cost-factor !!
  - A global estimate of project activities:

| | |
|---|---|
| Percentage of «Coding» ? | 15 - 20 % |
| Proportion of Validation et Verification ? | ~20% |
| All others : (Analysis,Design, Certification, Maintenance, Management). | 60 % |

  - These figures may vary substantially in particular industries (Automotive, Railways, Medical…)

## The Problem for Software-Quality

- A Very General Rule of Thumb:
  - Programming is not enough ! Overall,
    It is not even the most important cost-factor !!
  - A global estimate of project activities:

    Percentage of «Coding» ? 15 - 20 %
    Proportion of Validation et Verification ? ~20%
    All others : (Analysis,Design, Certification,
    Maintenance, Management). 60 %

  - These figures may vary substantially in particular industries (Automotive, Railways, Medical…)

## The Problem for Software-Quality

- A Very General Rule of Thumb:

  - Programming is not enough ! Overall,
    It is not even the most important cost-factor !!

  - A global estimate of project activities:

    | | |
    |---|---|
    | Percentage of «Coding» ? | 15 - 20 % |
    | Proportion of Validation et Verification ? | ~20% |
    | All others : (Analysis,Design, Certification, Maintenance, Management). | 60 % |

  - These figures may vary substantially in particular industries (Automotive, Railways, Medical…)

## The Problem for Software-Quality

- A Very General Rule of Thumb:
  - Programming is not enough ! Overall,
    It is not even the most important cost-factor !!
  - A global estimate of project activities:

    Percentage of «Coding» ? 15 - 20 %
    Proportion of Validation et Verification ? ~20%
    All others : (Analysis,Design, Certification,
    Maintenance, Management). 60 %
  - These figures may vary substantially in particular industries (Automotive, Railways, Medical…)

## The Problem for Software-Quality

- A Very General Rule of Thumb:

  - Programming is not enough ! Overall,
    It is not even the most important cost-factor !!

  - A global estimate of project activities:

    | | |
    |---|---|
    | Percentage of «Coding» ? | 15 - 20 % |
    | Proportion of Validation et Verification ? | ~20% |
    | All others : (Analysis,Design, Certification, Maintenance, Management). | 60 % |

  - These figures may vary substantially in particular industries (Automotive, Railways, Medical…)

## The Problem for Software-Quality

- A Very General Rule of Thumb:
  - Programming is not enough ! Overall,
    It is not even the most important cost-factor !!
  - A global estimate of project activities:

    | | |
    |---|---|
    | Percentage of «Coding» ? | 15 - 20 % |
    | Proportion of Validation et Verification ? | ~20% |
    | All others : (Analysis,Design, Certification, Maintenance, Management). | 60 % |

  - These figures may vary substantially in particular industries (Automotive, Railways, Medical…)

## The Problem for Software Quality:

## The Problem for Software Quality:

- There are various "pragmatic" ways to deal with software quality problems

## The Problem for Software Quality:

- There are various "pragmatic" ways to deal with software quality problems
  - Propaganda: all "real" (large) software has bugs … (That's true, but somewhat off the point)

# The Problem for Software Quality:

- There are various "pragmatic" ways to deal with software quality problems
  - Propaganda: all "real" (large) software has bugs … (That's true, but somewhat off the point)
  - Shifting the costs of bug :

## The Problem for Software Quality:

- There are various "pragmatic" ways to deal with software quality problems
  - Propaganda: all "real" (large) software has bugs … (That's true, but somewhat off the point)
  - Shifting the costs of bug :
    - reveal and fix it now, but ship later …
      or:

## The Problem for Software Quality:

- There are various “pragmatic” ways to deal with software quality problems
  - Propaganda: all “real” (large) software has bugs … (That’s true, but somewhat off the point)
  - Shifting the costs of bug :
    - reveal and fix it now, but ship later …
      or:
    - risking the cost of a legal battle
      (perhaps, in the future…)
      or

# The Problem for Software Quality:

- There are various “pragmatic” ways to deal with software quality problems
  - Propaganda: all “real” (large) software has bugs … (That’s true, but somewhat off the point)
  - Shifting the costs of bug :
    - reveal and fix it now, but ship later …
      or:
    - risking the cost of a legal battle
      (perhaps, in the future…)
      or
    - risking potential damage to the brand-image
      (difficult to evaluate, and often considered irreal)

## The Problem for Software Quality:

- There are various “pragmatic” ways to deal with software quality problems
  - Propaganda: all “real” (large) software has bugs … (That’s true, but somewhat off the point)
  - Shifting the costs of bug :
    - reveal and fix it now, but ship later …
      or:
    - risking the cost of a legal battle
      (perhaps, in the future…)
      or
    - risking potential damage to the brand-image
      (difficult to evaluate, and often considered irreal)

## The Problem for Software Quality:

## The Problem for Software Quality:

- on the other side – you can't test infinitely,
  and thorough verification techniques
  are unfortunately ways more costly than shallow testing !

## The Problem for Software Quality:

## The Problem for Software Quality:

? ? ?

## The Problem for Software Quality:

## The Problem for Software Quality:

- … these are in practice often the real questions for the management of a software process …

## Why is it important to get software right?

## Why is it important to get software right?

- One common answer:

  Since information technology becomes more and more pervasive, Reliability, Safety and Security become more critical

## Why is it important to get software right?

- One common answer:

  Since information technology becomes more and more pervasive, Reliability, Safety and Security become more critical

  - Consider the application domains with obvious criticality:
    - transport systems (Cars, Métros, TGV), aviation controls, aerospace, ...
    - critical industrial processes, nuclear power plants, weapons, ...
    - medical technologies: tele-surgery, radiation control...
    - critical telecommunication infrastructures and networks,
    - electronic commerce

## Why is it important to get software right?

## Why is it important to get software right?

- For most of them exist certification processes, legal requirements, etcpp.

## Why is it important to get software right?

## Why is it important to get software right?

- This should be the most important reason, but, actually, it isn't.

## Why is it important to get software right?

## Why is it important to get software right?

- The complexity of large software-projects can simply not be mastered without advanced software engineering techniques …

## What are the Sub-disciplines of SE

## What are the Sub-disciplines of SE

So ...

## What are the Sub-disciplines of SE

So ...

- What is Software Engineering (SE) as a discipline about ?

## What are the Sub-disciplines of SE

So ...

- What is Software Engineering (SE) as a discipline about ?
- What are the sub-disciplines ?

## What are the Sub-disciplines of SE

## What are the Sub-disciplines of SE

- Well - again common knowledge [thanks wikipedia!]

## What are the Sub-disciplines of SE

- Well - again common knowledge [thanks wikipedia!]
  - *[1] Requirements engineering: The elicitation, analysis, specification, and validation of requirements for software.*

## What are the Sub-disciplines of SE

- Well - again common knowledge [thanks wikipedia!]
  - *[1] Requirements engineering: The elicitation, analysis, specification, and validation of requirements for software.*
  - *[2] Software design: The process of defining the architecture, components, interfaces, and other characteristics of a system or component. It is also defined as the result of that process.*

# What are the Sub-disciplines of SE

- Well - again common knowledge [thanks wikipedia!]
  - *[1] Requirements engineering: The elicitation, analysis, specification, and validation of requirements for software.*
  - *[2] Software design: The process of defining the architecture, components, interfaces, and other characteristics of a system or component. It is also defined as the result of that process.*
  - *[3] Software construction: The detailed creation of working, meaningful software through a combination of coding, verification, testing and debugging.*

## What are the Sub-disciplines of SE

- Well - again common knowledge [thanks wikipedia!]
  - *[1] Requirements engineering: The elicitation, analysis, specification, and validation of requirements for software.*
  - *[2] Software design: The process of defining the architecture, components, interfaces, and other characteristics of a system or component. It is also defined as the result of that process.*
  - *[3] Software construction: The detailed creation of working, meaningful software through a combination of coding, verification, testing and debugging.*
  - ...

## What are the Sub-disciplines of SE

## What are the Sub-disciplines of SE

- Well - again common knowledge [thanks wikipedia!]

## What are the Sub-disciplines of SE

- Well - again common knowledge [thanks wikipedia!]
  - . . .
  - *[4] Software verification: The verification of the behavior of a program on a finite set of test cases, suitably selected from the usually infinite executions domain, against the expected behavior.*

    [This may include simulation, animation, test-generation, formal proof and model-checking activities ... ]

## What are the Sub-disciplines of SE

- Well - again common knowledge [thanks wikipedia!]
  - . . .
  - *[4] Software verification: The verification of the behavior of a program on a finite set of test cases, suitably selected from the usually infinite executions domain, against the expected behavior.*

    [This may include simulation, animation, test-generation, formal proof and model-checking activities ... ]
  - *[5] Software maintenance: The totality of activities required to provide cost-effective support to software.*
  - ...

## What are the Sub-disciplines of SE

## What are the Sub-disciplines of SE

- Well - again common knowledge [thanks wikipedia!]
  - . . .

## What are the Sub-disciplines of SE

- Well - again common knowledge [thanks wikipedia!]
  - . . .
  - *[6] Software configuration management: The identification of the configuration of a system at distinct points in time for the purpose of systematically controlling changes to the configuration, and maintaining the integrity and traceability of the configuration throughout the system life cycle.*

# What are the Sub-disciplines of SE

- Well - again common knowledge [thanks wikipedia!]
  - *. . .*
  - *[6] Software configuration management: The identification of the configuration of a system at distinct points in time for the purpose of systematically controlling changes to the configuration, and maintaining the integrity and traceability of the configuration throughout the system life cycle.*
  - *[7] Software engineering management: The application of management activities—planning, coordinating, measuring, monitoring, controlling, and reporting—to ensure that the development and maintenance of software is systematic, disciplined, and quantified.*

    [Again: this is not what we do in this course: it requires more experience and a concrete process to do this ...]
  - . . .

## What are the Sub-disciplines of SE

## What are the Sub-disciplines of SE

- Well - again common knowledge [thanks wikipedia!]
    - . . .

## What are the Sub-disciplines of SE

- Well - again common knowledge [thanks wikipedia!]
  - *. . .*
  - *[8] Software engineering process: The definition, implementation, assessment, measurement, management, change, and improvement of the software life cycle process itself.*

## What are the Sub-disciplines of SE

- Well - again common knowledge [thanks wikipedia!]
  - *. . .*
  - *[8] Software engineering process: The definition, implementation, assessment, measurement, management, change, and improvement of the software life cycle process itself.*
  - *[9] Software engineering tools and methods: The computer-based tools that are intended to assist the software life cycle processes (see Computer-aided software engineering) and the methods which impose structure on the software engineering activity with the goal of making the activity systematic and ultimately more likely to be successful.*

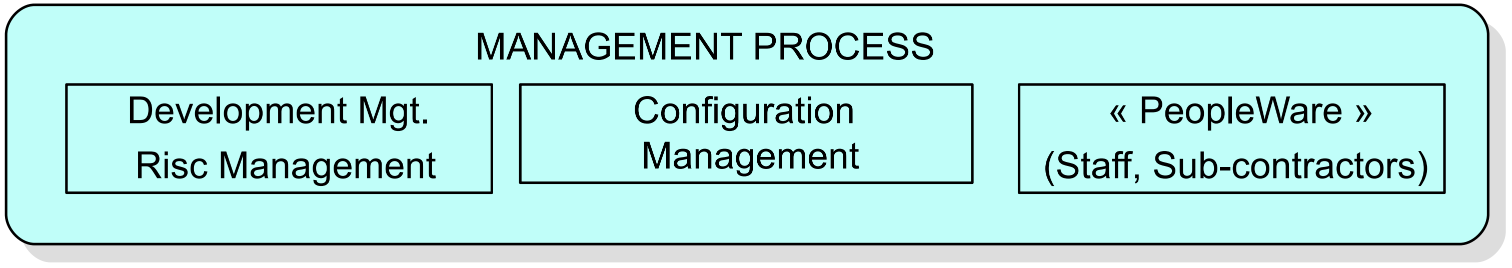
## What are the Sub-disciplines of SE

- Well - again common knowledge [thanks wikipedia!]
  - *. . .*
  - *[8] Software engineering process: The definition, implementation, assessment, measurement, management, change, and improvement of the software life cycle process itself.*
  - *[9] Software engineering tools and methods: The computer-based tools that are intended to assist the software life cycle processes (see Computer-aided software engineering) and the methods which impose structure on the software engineering activity with the goal of making the activity systematic and ultimately more likely to be successful.*
  - *[10] Software quality management: The degree to which a set of inherent characteristics fulfils requirements.*

## A "Software Engineering Process" (example)

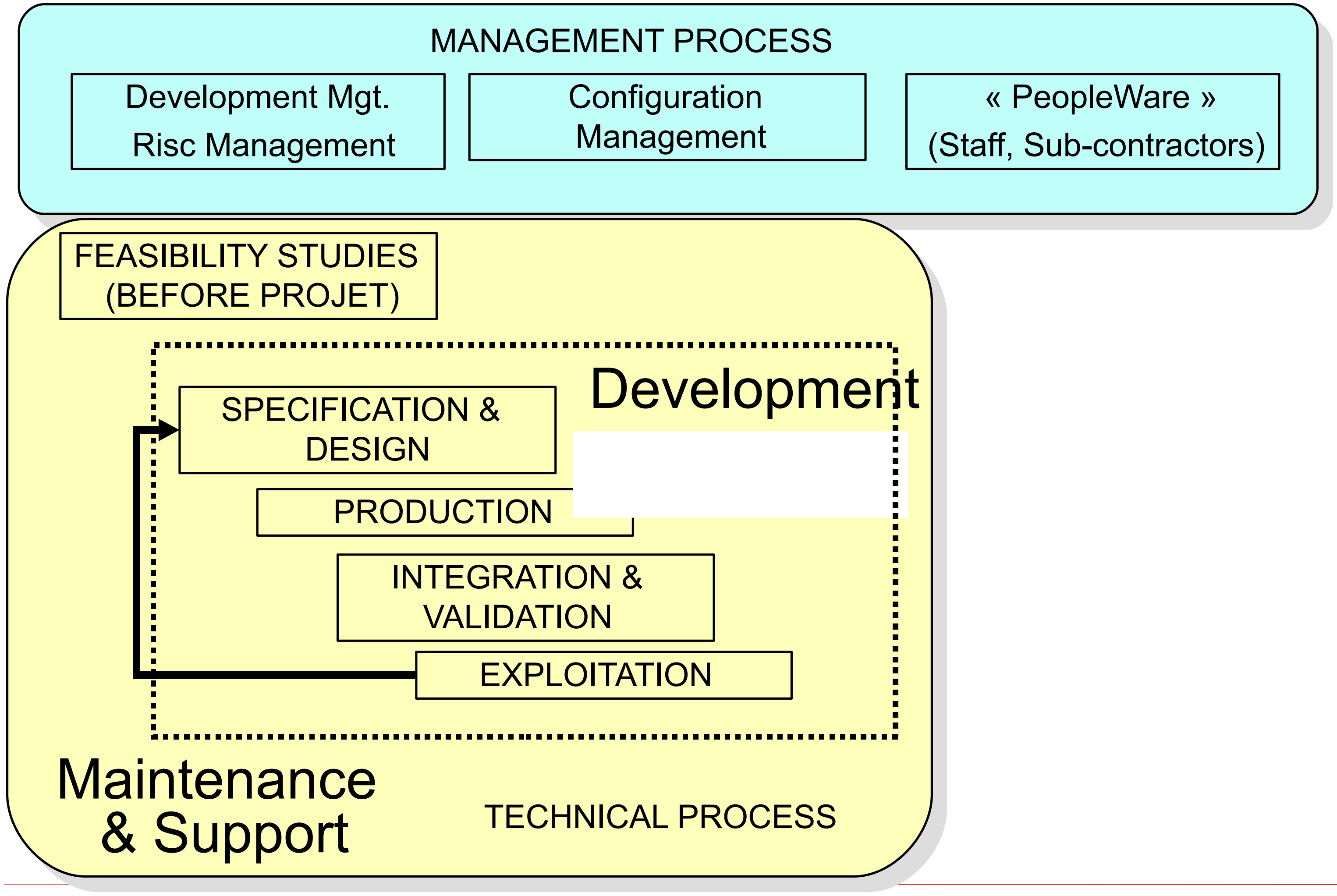
TECHNICAL PROCESS

# A "Software Engineering Process" (example)

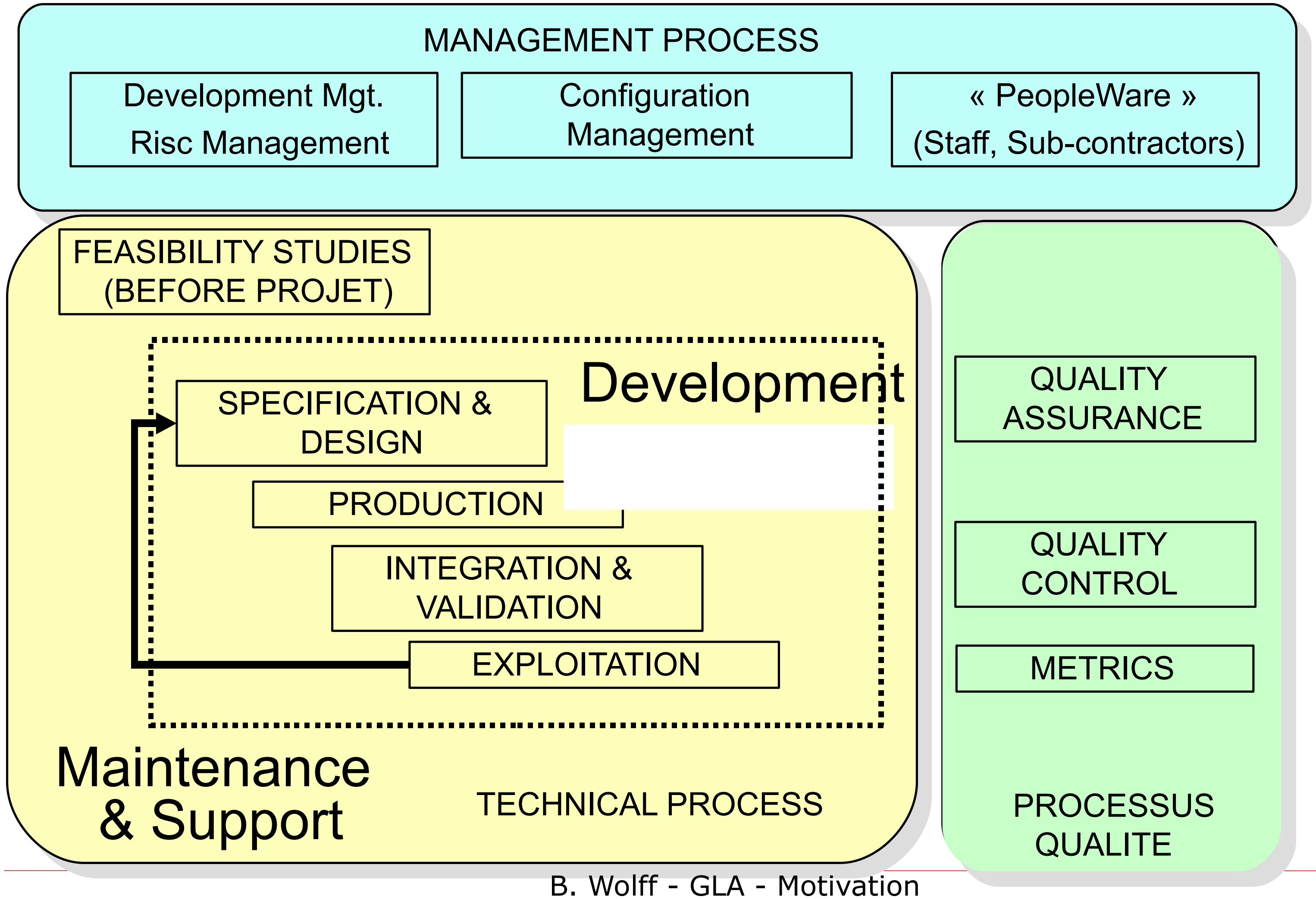


TECHNICAL PROCESS

# A “Software Engineering Process” (example)

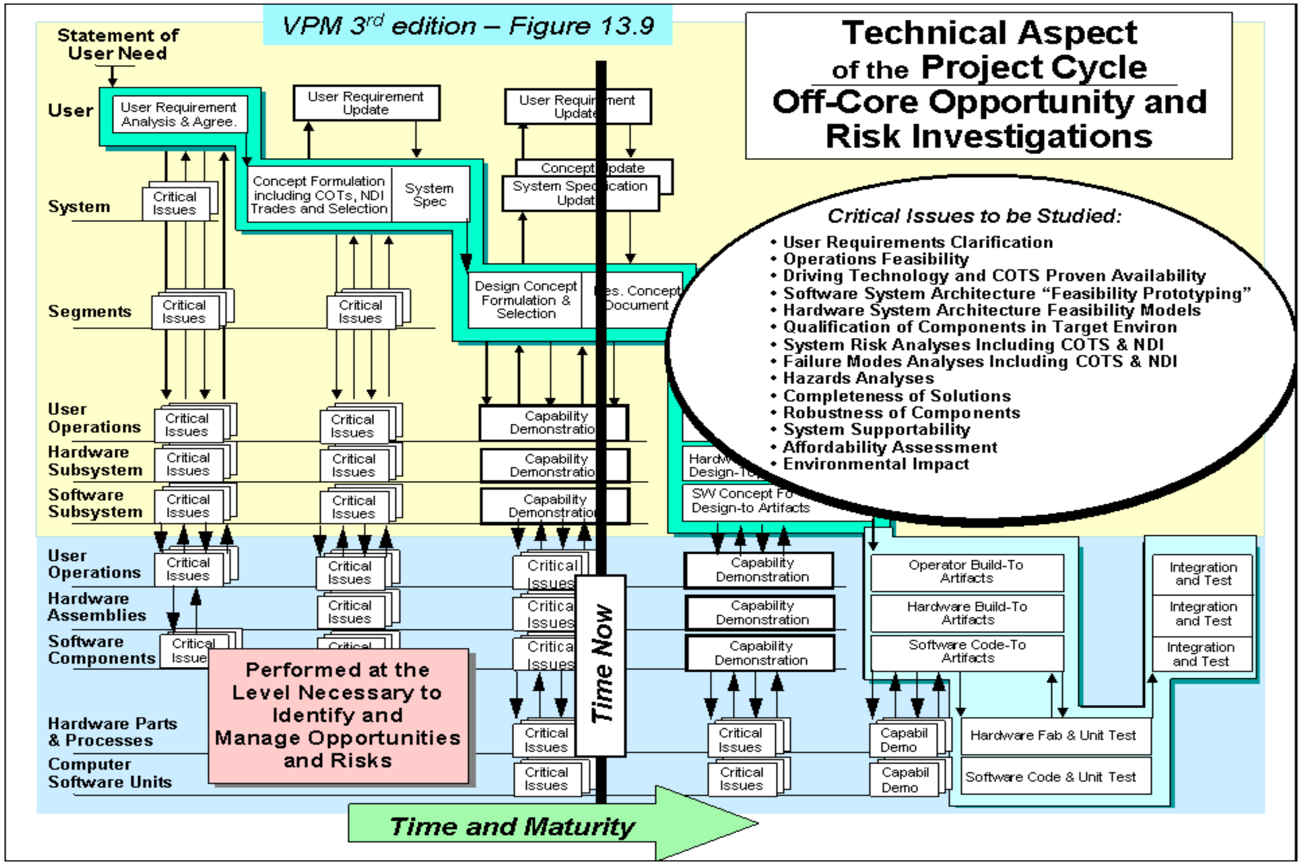


# A "Software Engineering Process" (example)



# A “Software Engineering Process” (example)

- Another Example: The VPM3-Model (Daimler)


VPM 3rd edition – Figure 13.9
Technical Aspect of the Project Cycle
Off-Core Opportunity and Risk Investigations
Statement of User Need
User
System
Segments
User Operations
Hardware Subsystem
Software Subsystem
User Operations
Hardware Assemblies
Software Components
Hardware Parts & Processes
Computer Software Units
User Requirement Analysis & Agree.
User Requirement Update
Concept Formulation including COTs, NDI Trades and Selection
System Spec
Concept Update
System Specification Update
Design Concept Formulation & Selection
Des. Concept Document
Critical Issues
Capability Demonstration
SW Concept For Design-to Artifacts
Operator Build-To Artifacts
Hardware Build-To Artifacts
Software Code-To Artifacts
Integration and Test
Capabil Demo
Hardware Fab & Unit Test
Software Code & Unit Test
Time Now
Performed at the Level Necessary to Identify and Manage Opportunities and Risks
Time and Maturity
Critical Issues to be Studied:
• User Requirements Clarification
• Operations Feasibility
• Driving Technology and COTS Proven Availability
• Software System Architecture “Feasibility Prototyping”
• Hardware System Architecture Feasibility Models
• Qualification of Components in Target Environ
• System Risk Analyses Including COTS & NDI
• Failure Modes Analyses Including COTS & NDI
• Hazards Analyses
• Completeness of Solutions
• Robustness of Components
• System Supportability
• Affordability Assessment
• Environmental Impact

## How can software be «built systematically»?

## How can software be «built systematically»?

- Organise a development into formally described process !

## How can software be «built systematically»?

- Organise a development into formally described process !
  - ... with identified phases, (which correspond partly to the aforementioned “SE disciplines” )

## How can software be «built systematically»?

- Organise a development into formally described process !
  - ... with identified phases, (which correspond partly to the aforementioned “SE disciplines” )
  - ... staff (and organisation and cost-plans)

## How can software be «built systematically»?

- Organise a development into formally described process !
  - ... with identified phases, (which correspond partly to the aforementioned “SE disciplines” )
  - ... staff (and organisation and cost-plans)
  - ... defined deliverables (i.e. documents, codes, ... )

## How can software be «built systematically»?

- Organise a development into formally described process !
  - ... with identified phases, (which correspond partly to the aforementioned "SE disciplines" )
  - ... staff (and organisation and cost-plans)
  - ... defined deliverables (i.e. documents, codes, ... )
  - ... procedures (and tools !) to validate the quality of the deliverables (reviews, static checks)

## How can software be «built systematically»?

- Organise a development into formally described process !
  - ... with identified phases, (which correspond partly to the aforementioned "SE disciplines" )
  - ... staff (and organisation and cost-plans)
  - ... defined deliverables (i.e. documents, codes, ... )
  - ... procedures (and tools !) to validate the quality of the deliverables (reviews, static checks)
  - ... procedures to version and configure deliverables (in particular code)

## How can software be «built systematically»?

- Organise a development into formally described process !
  - ... with identified phases, (which correspond partly to the aforementioned "SE disciplines" )
  - ... staff (and organisation and cost-plans)
  - ... defined deliverables (i.e. documents, codes, ... )
  - ... procedures (and tools !) to validate the quality of the deliverables (reviews, static checks)
  - ... procedures to version and configure deliverables (in particular code)
  - Compare: THE SWEBOOK

    IEEE Computer Society an international standard ISO/IEC TR 19759:2005

## How can software be «built systematically»?

## How can software be «built systematically»?

## How can software be «built systematically»?

- Let's have a close look into another Example Process-Model: The V-Model. [German Administration 2005]
  http://de.wikipedia.org/wiki/V-Modell_(Entwicklungsstandard)#cite_note-6
  It identifies:

## How can software be «built systematically»?

- Let's have a closer look into another Example Process-Model: The V-Model. [German Administration 2005]
  http://de.wikipedia.org/wiki/V-Modell_(Entwicklungsstandard)#cite_note-6
  It identifies:
  - ... phases : Requirement, Architectural Design, Design, Code, Tests, passed deployments, ...

## How can software be «built systematically»?

- Let's have a closer look into another Example Process-Model: The V-Model. [German Administration 2005]
  http://de.wikipedia.org/wiki/V-Modell_(Entwicklungsstandard)#cite_note-6
  It identifies:
  - ... phases : Requirement, Architectural Design, Design, Code, Tests, passed deployments, ...
  - ... defined deliverables as milestones (i.e. documents, codes, ... ) based on templates to be "taylored" to the task

## How can software be «built systematically»?

- Let's have a close look into another Example Process-Model: The V-Model. [German Administration 2005]
  http://de.wikipedia.org/wiki/V-Modell_(Entwicklungsstandard)#cite_note-6
  It identifies:
  - ... phases : Requirement, Architectural Design, Design, Code, Tests, passed deployments, ...
  - ... defined deliverables as milestones (i.e. documents, codes, ... ) based on templates to be "taylored" to the task
  - ... tools & procedures : syntax-based editors, IDEs, version & configuration management and an access control management for the various different roles in the process, ...

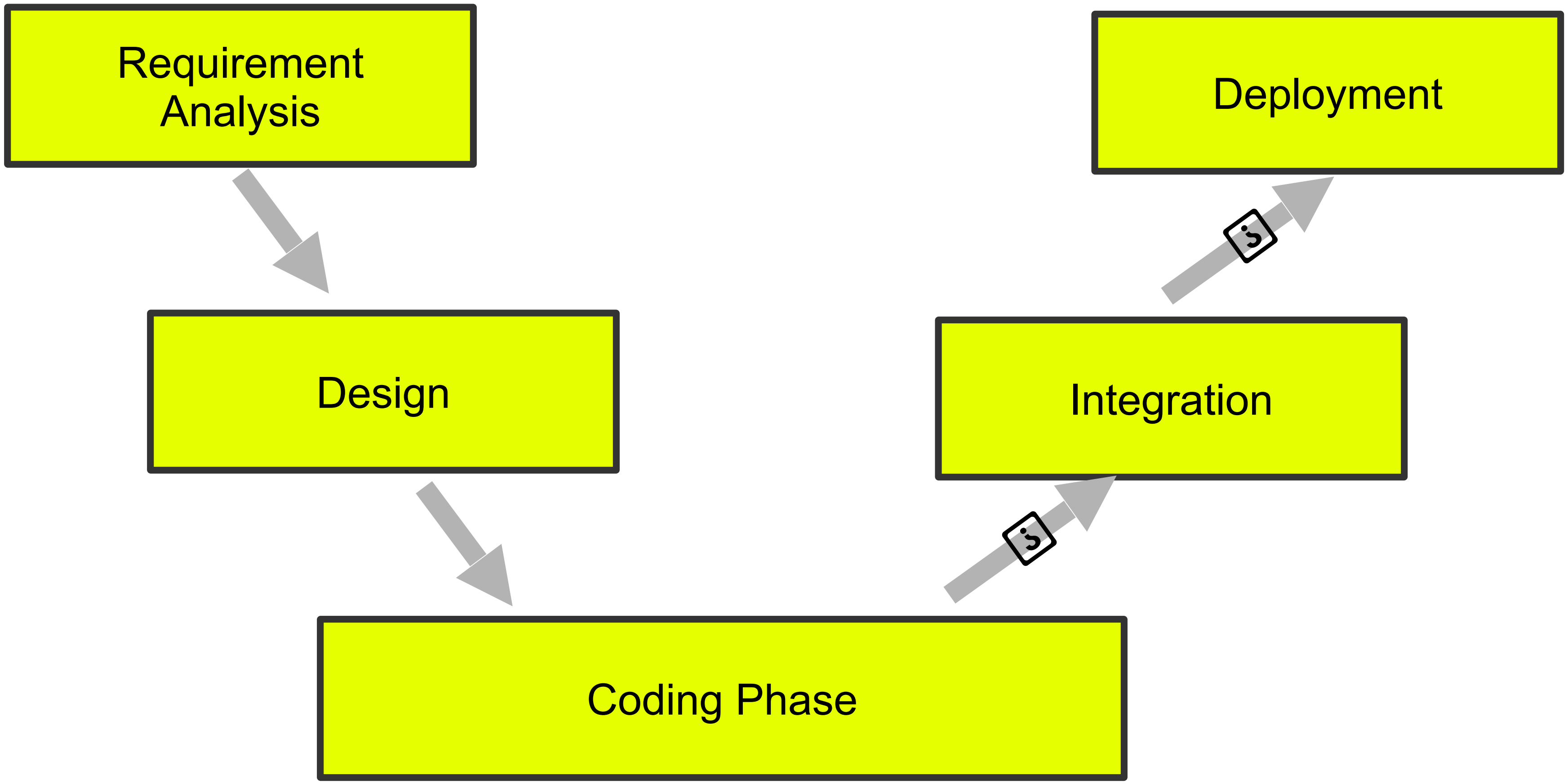
# How can software be «built systematically»?

- Let's have a closer look into another Example Process-Model: The V-Model. [German Administration 2005]
  http://de.wikipedia.org/wiki/V-Modell_(Entwicklungsstandard)#cite_note-6
  It identifies:
  - ... phases : Requirement, Architectural Design, Design, Code, Tests, passed deployments, ...
  - ... defined deliverables as milestones (i.e. documents, codes, ... ) based on templates to be "taylored" to the task
  - ... tools & procedures : syntax-based editors, IDEs, version & configuration management and an access control management for the various different roles in the process, ...

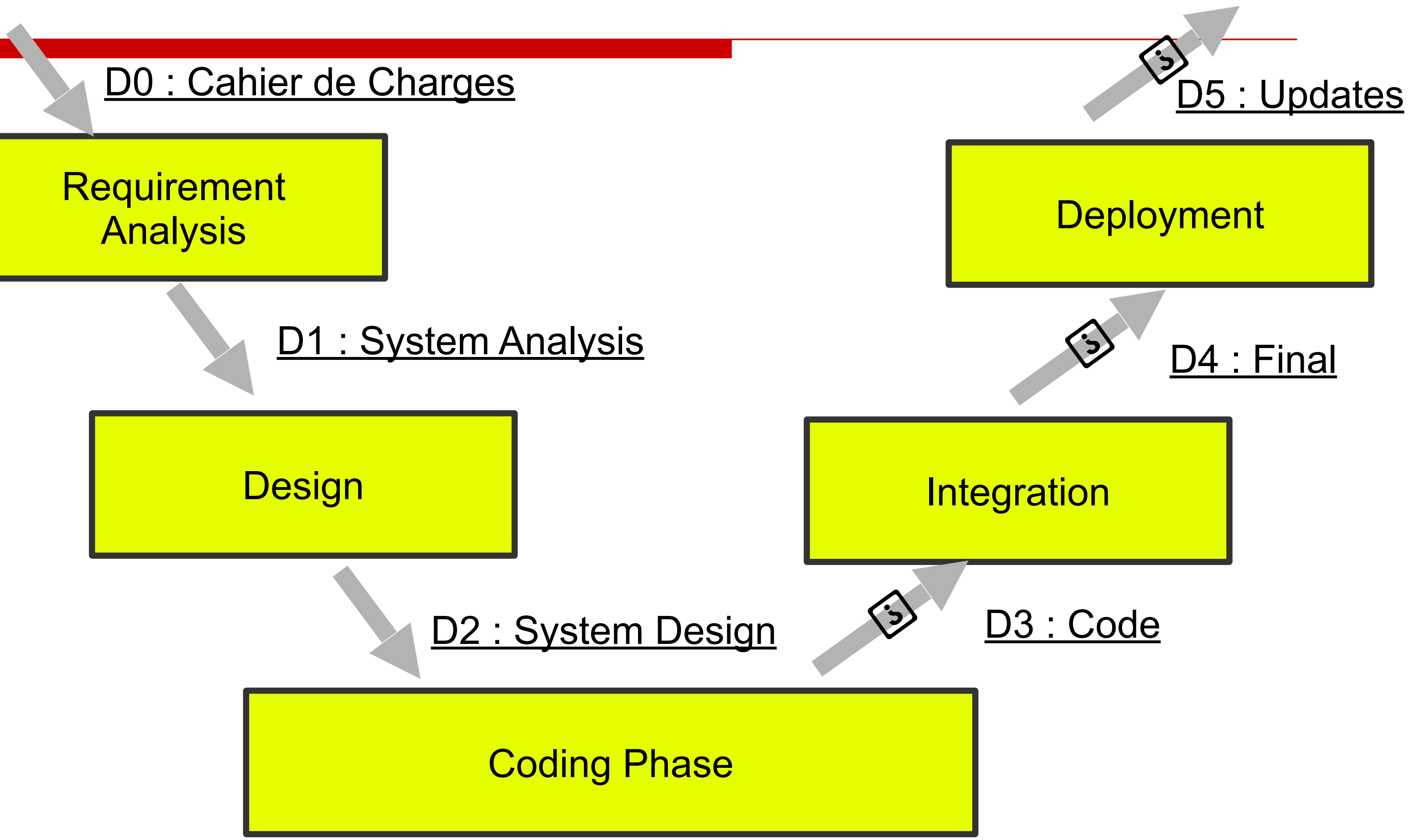
## How can software be «built systematically»?

- Let's have a closer look into another Example Process-Model: The V-Model. [German Administration 2005]
  http://de.wikipedia.org/wiki/V-Modell_(Entwicklungsstandard)#cite_note-6
  It identifies:
  - ... phases : Requirement, Architectural Design, Design, Code, Tests, passed deployments, ...
  - ... defined deliverables as milestones (i.e. documents, codes, ... ) based on templates to be "taylored" to the task
  - ... tools & procedures : syntax-based editors, IDEs, version & configuration management and an access control management for the various different roles in the process, ...

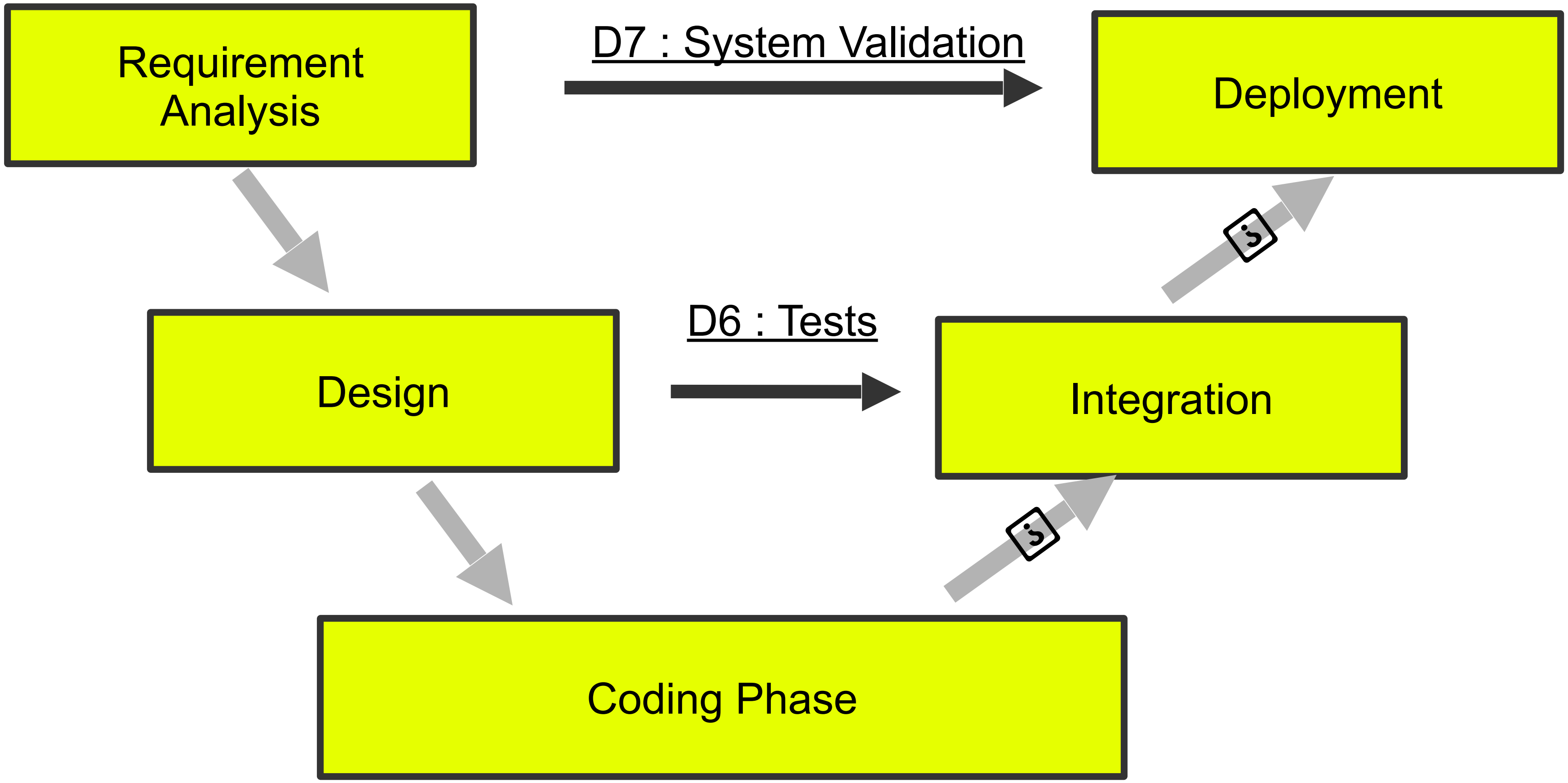
# A Schematic View on the V-Model



# A Schematic View on the V-Model



# A Schematic View on the V-Model



## The V-Model

## The V-Model

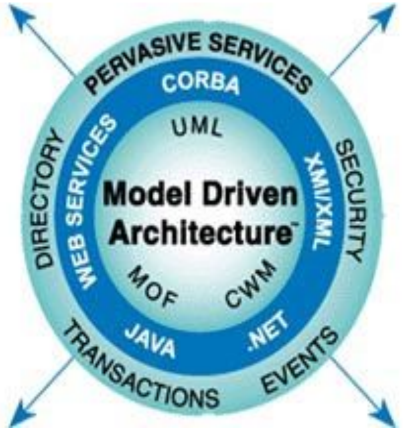
- We will use the V-Model as a kind of "typical classical process-model"

  Many processes are just a variant of it.

- However, there is no such thing like "the process" in industry,

- … et chacun fait ca a sa sauce …

## Alternatives to the V-Model



- IBM Rational Unified Process (RUP)

- Idea : Using UML and OCL integrated into the Deliverables (documents)
- Idea : Allows for semi-formal editing, more precise notation and therefore better communication
- Analysis, Design and Code Documents CONTAIN standardised diagrammatic specification elements (the "model") which can be automatically validated
- Code and Tests can partially be generated from design models (Model-Driven Engineering (MDA))

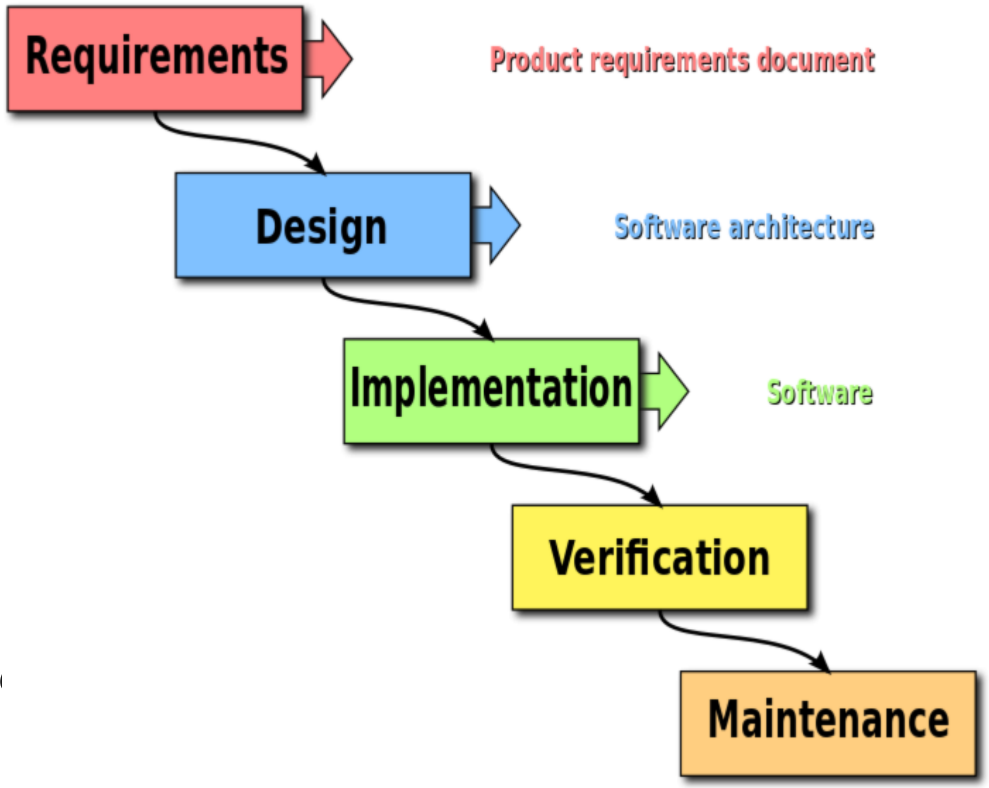
# Alternatives to the V-Model

Waterfall Model

[Benington 56, Royce 70]

Royce presented this model as an example of a flawed, non-working models.

Category: Academic example.
Never works like this in practi

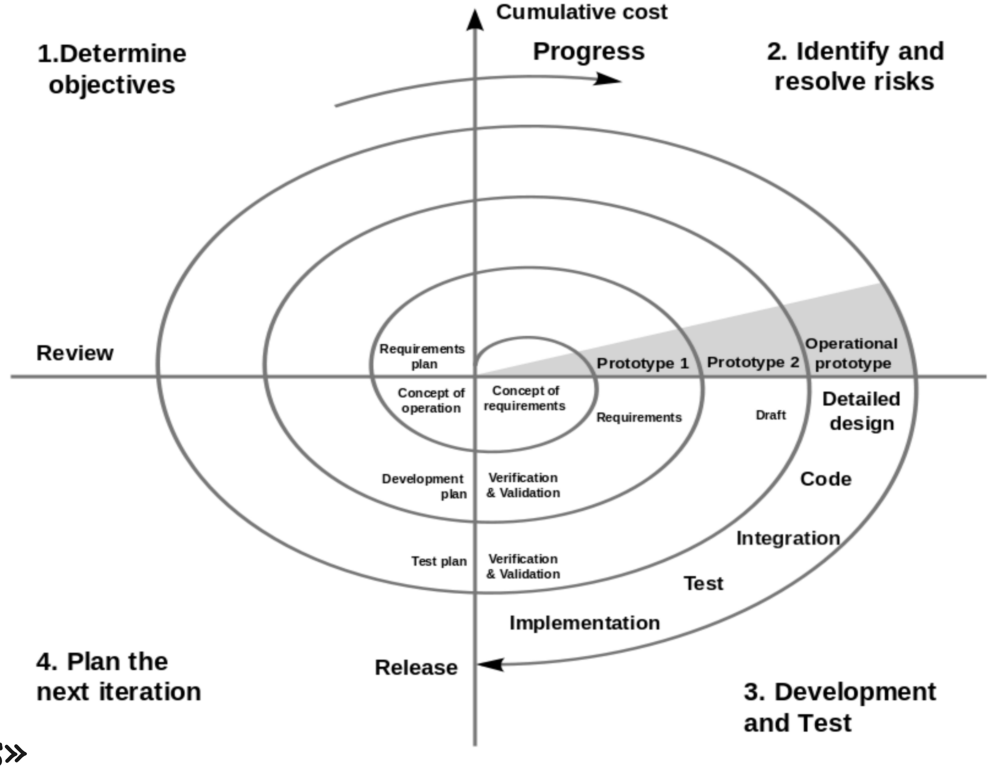


# Alternatives to the V-Model

- ## Spiral Model [Barry Boehm 88]

combines some key aspect of the waterfall model and rapid prototyping methodologies, in an effort to combine advantages of top-down and bottom-up concepts.

Today mostly a conceptual reference; ideas are retaken in « agile development processes»



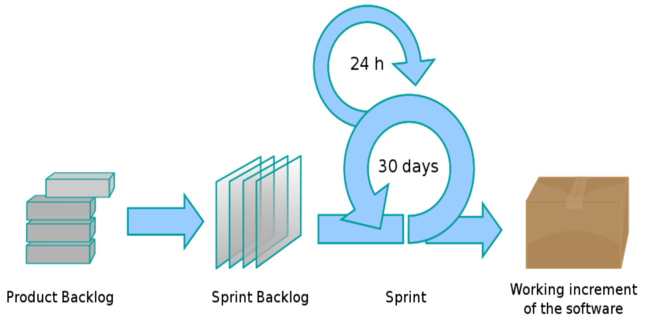
# Alternatives to the V-Model

- Agile Software Development

  [Beck et al 2001, V2: 2010]

  AD advocates:

  - adaptive planning,
  - evolutionary, incremental development,
  - early delivery,
  - continuous improvement,
  - and it encourages rapid and flexible response to change

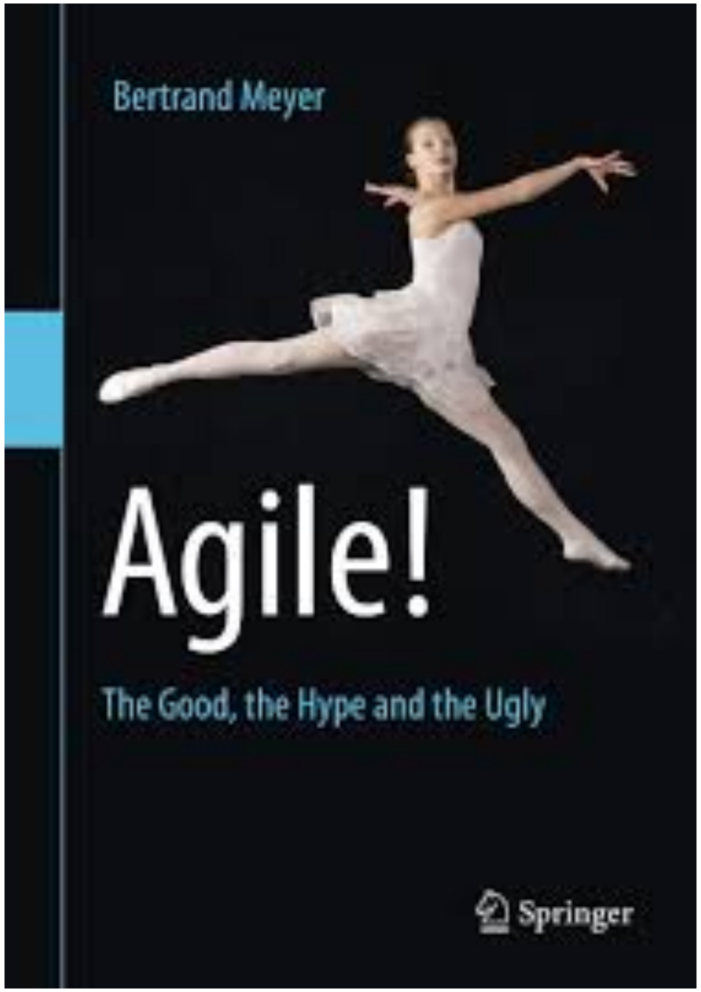


Particular variants are called « Extreme Programming » (with an emphasis on early, handwritten tests)

SCRUM (with emphasis on social organisation and continuous team-reviews)

# Alternatives to the V-Model

- An amusing book analysing and criticising Agile Methods by one of the Peers of Software Engineering is :


Bertrand Meyer
Agile!
The Good, the Hype and the Ugly
Springer