Engenharia de Software

José Cascais Brás

Módulo 1

Learning Objectives

PROFESSIONAL SOFTWARE DEVELOPMENT

What is meant by software engineering.

SOFTWARE ENGINEERING ETHICS

A brief introduction to ethical issues that affect software engineering.

CASE STUDIES

An introduction to four examples that are used in later topics.

KEY POINTS

What is important to retain

SOFTWARE ENGINEERING

“Today's world could not live without software”

Sommerville, 2011

SOFTWARE

Everything around us involves software:

- From electronic products, transportation services, medical, telecommunications, military, industrial and financial, entertainment, education, etc.

SOFTWARE

What is software?

What is software?

- More than source programs and executables: it also comprises the associated documentation, data and settings.
- All of this impacts the future evolution and maintenance of the software.

SOFTWARE

The law of unintended consequences

Pressman, 2006

"Have you ever noticed how the invention of one technology can have profound unexpected effects on other seemingly unrelated technologies, on commercial enterprises, on people, and even on culture as a whole?"

The law states that "while observing the problem in a deep way, it seems to be clear and a solution also seems to sure-fire. But at the same time the solution also creates another problem".

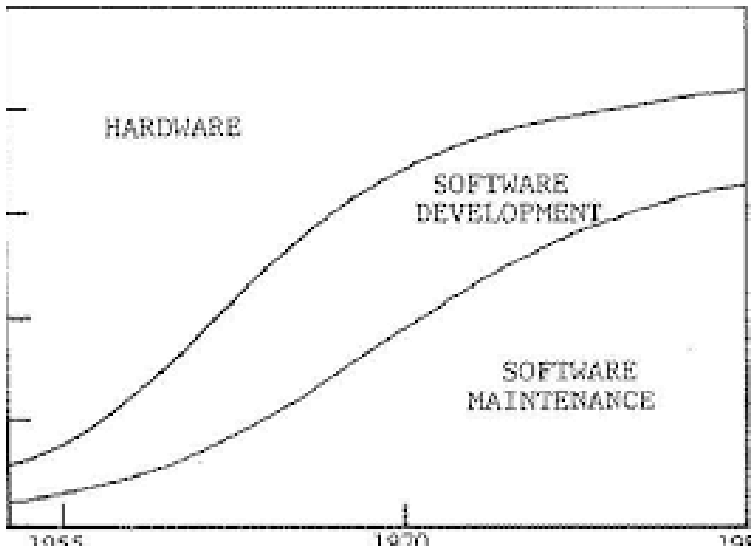
In the 50's it was not expected that the software would take the current proportions.

What are the consequences of this?

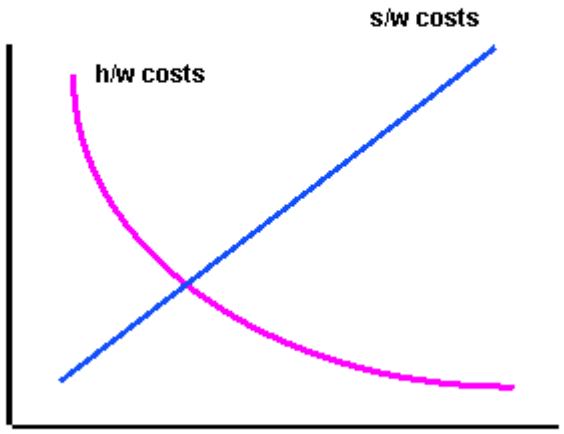
SOFTWARE PROBLEMS

- **Software that doesn't work**
- **High costs**
- **Deadlines not met**
- **Low quality:**
  - full of defects
  - dubious reliability
  - hard to use
  - slow
  - not portable

SOFTWARE COSTS

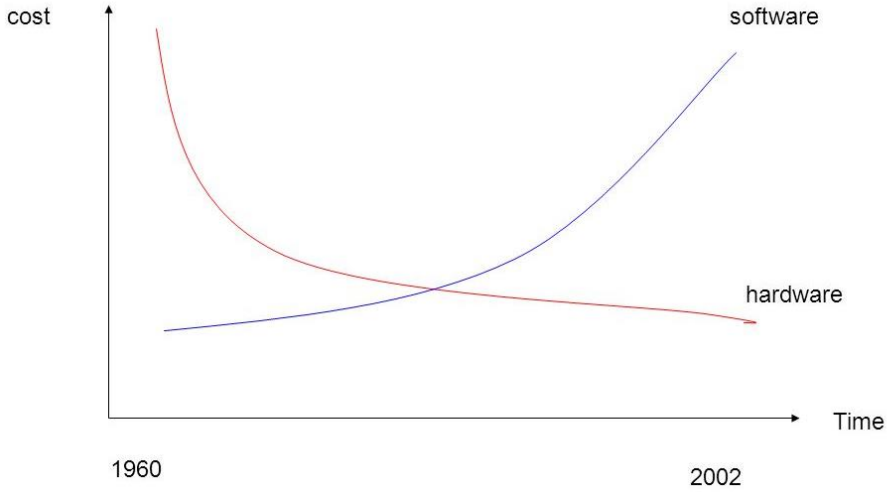


Hardware Costs vs Software Costs
(% of overall costs)



Dillon : CSE470 : INTRO 5

Hardware vs Software



SOFTWARE COSTS

Software costs

Often dominate **computer system** costs.

The costs of software on a PC are often greater than the hardware cost.

Software Costs

Costs more to maintain than it does to develop.

For systems with a long life, maintenance costs may be several times development costs.

Software Engineering

Is concerned with cost effective software development.

SOFTWARE PROJECT FAILURE

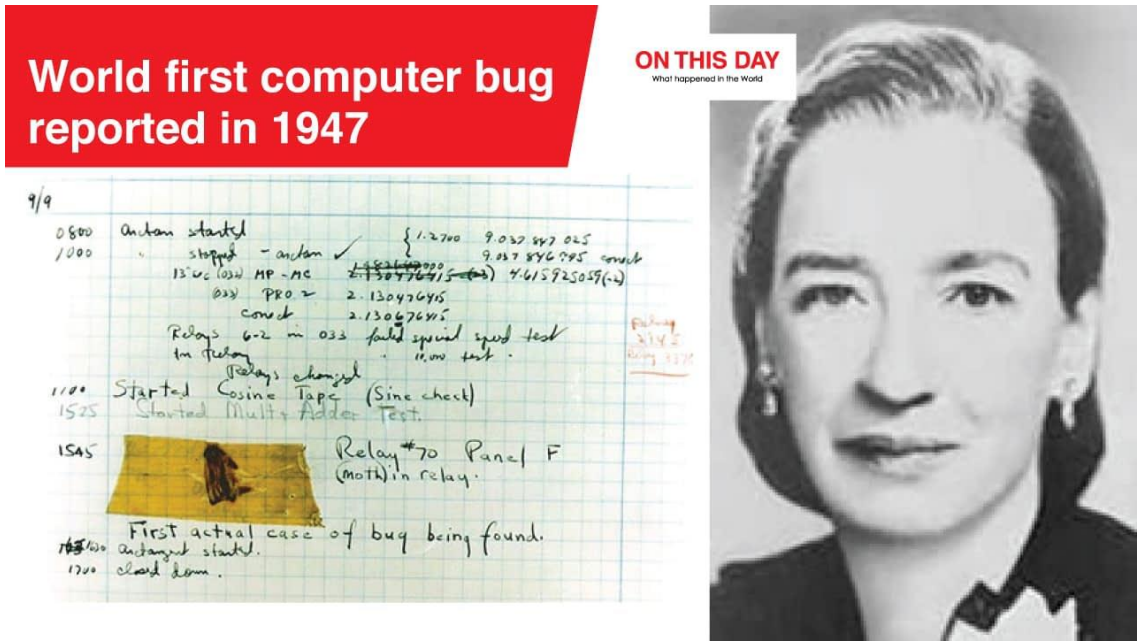
1. Increasing system complexity

As new software engineering techniques help us to build larger, more complex systems, the demands change. Systems have to be built and delivered more quickly; larger, even more complex systems are required; systems have to have new capabilities that were previously thought to be impossible.

2. Failure to use software engineering methods

It is fairly easy to write computer programs without using software engineering methods and techniques. Many companies have drifted into software development as their products and services have evolved. They do not use software engineering methods in their everyday work. Consequently, their software is often more expensive and less reliable than it should be.

BUGS



On September 9, 1947, a team of computer scientists and engineers reported the world's first computer bug. A bug is a flaw or glitch in a system. Thomas Edison reported "bugs" in his designs as early as the 1800s, but this was the first bug identified in a computer. Today, software bugs can impact the functioning, safety, and security of computer operating systems. "Debugging" and bug management are important parts of the computer science industry.

This bug, however, was literally a bug. "First actual case of bug being found," one of the team members wrote in the logbook. The team at Harvard University in Cambridge, Massachusetts, found that their computer, the Mark II, was delivering consistent errors. When they opened the computer's hardware, they found ... a moth. The trapped insect had disrupted the electronics of the computer.

Computer Bug
"First actual case of bug being found," according to the brainiacs at Harvard, 1945. The engineers who found the moth were the first to literally "debug" a machine.

HISTORICAL BUGS

Mariner I – 1962

– Mission to observe planet Venus

– Mathematical formula was mistakenly transcribed to the computer

–Drifted off course and destroyed 4 min after launch

–Loss: US$ 18.5 million

HISTORICAL BUGS

Floating point division on Intel Pentium processors – 1993

– Error in divisions within a range of numbers (~0.006% error in rounding)

– 3 to 5 million defective parts

– Recall to everyone who wanted to trade

– Cost Intel $475 million

$$\frac{4195835}{3145727} = 1.3338204491362410 02$$

$$\frac{4195835}{3145727} = 1.3337390689020375 89$$

HISTORICAL BUGS

Ariane 5 flight 501 – 1996

– It took a decade to develop and cost 7 billion dollars.

– Rocket with reused code from Ariane 4 (other hardware);

– Integer overflow: conversion from 64-bit float to 16-bit signed integer;

– The rocket's primary processor overloaded the engines which disintegrated in 40 seconds;

– Unmanned (no casualties); loss of US$ 370 million

HISTORICAL BUGS

Millennium Bug (Y2K) – 2000

– Dates with only 2 digits for the year

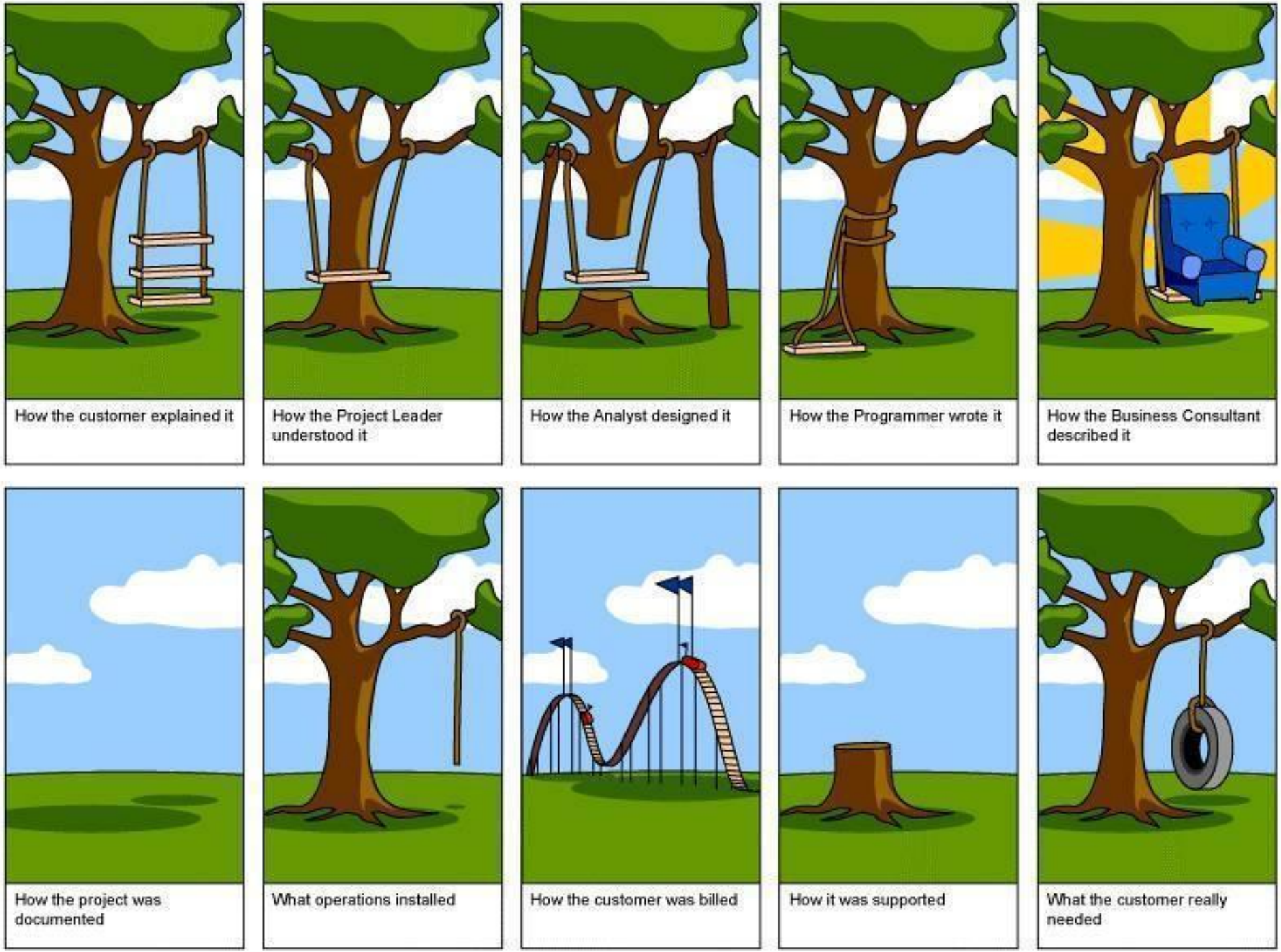
– One of the greatest hysterias in history

– At the turn of the year 2000, the concern was that it would count as 1900

– Between US$300 and US$500 billion worldwide

Why so many bugs occur?

- Lack of tests?
- Poorly made projects?
- Lack of quality control?
- Does the customer not know what he wants?
- Developers do not understand what the user wants?


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

SOFTWARE ENGINEERING

PROFESSIONAL SOFTWARE DEVELOPMENT

SOFTWARE ENGINEERING

A set of processes, methods, techniques and tools that help produce higher quality software at lower cost.

Include development process and project management activities

SOFTWARE ENGINEERING

Some important milestones

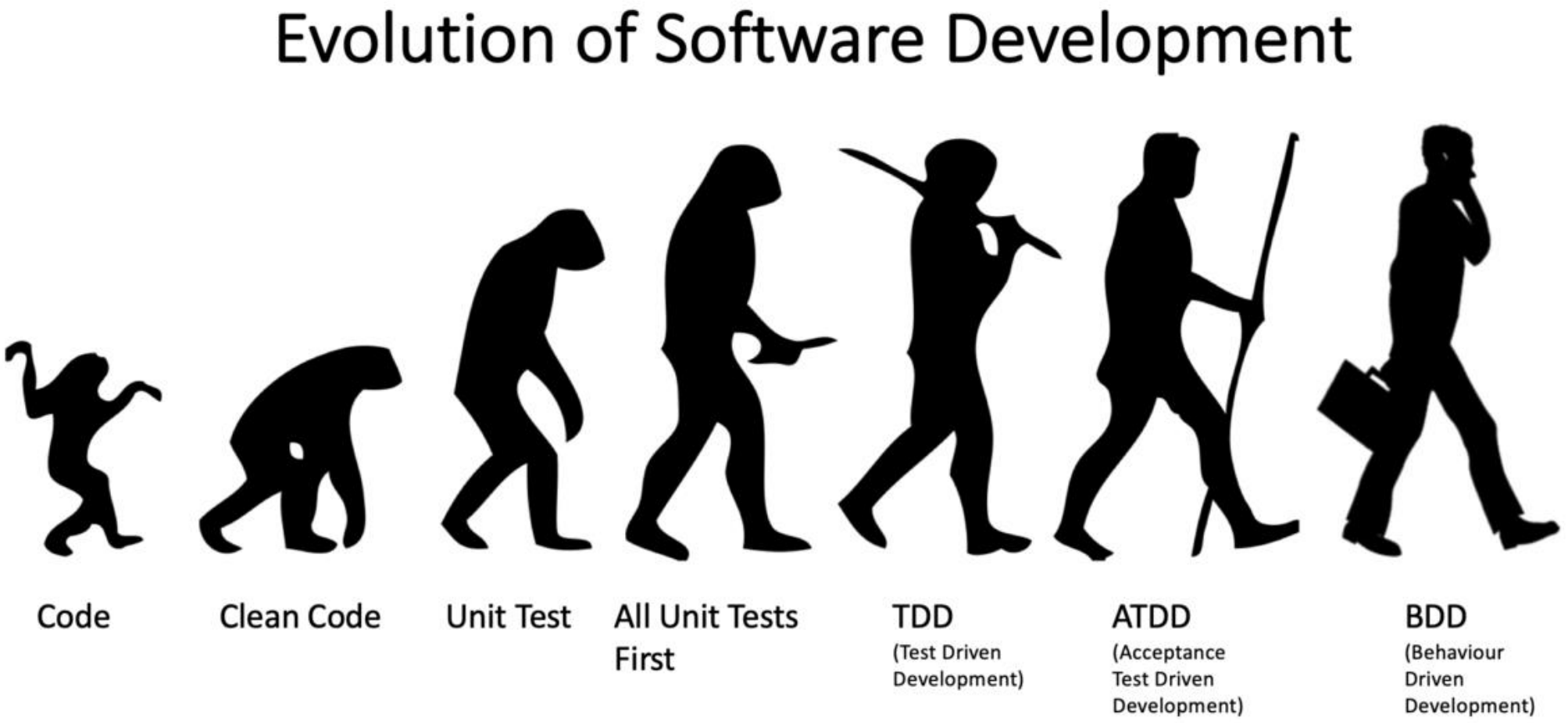
YEAR	FACTS
1950 to 1968	early times: ad hoc development
1968	Software Crisis • Software Engineering term officially defined
1970	Waterfall Model (Royce)
1974	Structured analysis/design (Stevens, Myers, Constantine, de Marco and Yourdon)
1975	The Mythical Man-Month (Fred Brooks)
1975	1st National Conference on Software Engineering (Washington DC) - became International in 1976 (ICSE)
1980 to 2000	Development of the various areas of ES (e.g. project planning, software estimation (COCOMO), risk management, process models and quality models, software testing, dev tools and environments, etc.)
Late 1980s	Emergence of object-oriented analytics
1997	UML (Unified Modeling Language)
2001	The Agile Manifesto
2002 onwards	multiple agile processes, focus on customer interaction and ease of evolution

SOFTWARE ENGINEERING (1950 to 1968)

Early times: ad hoc development

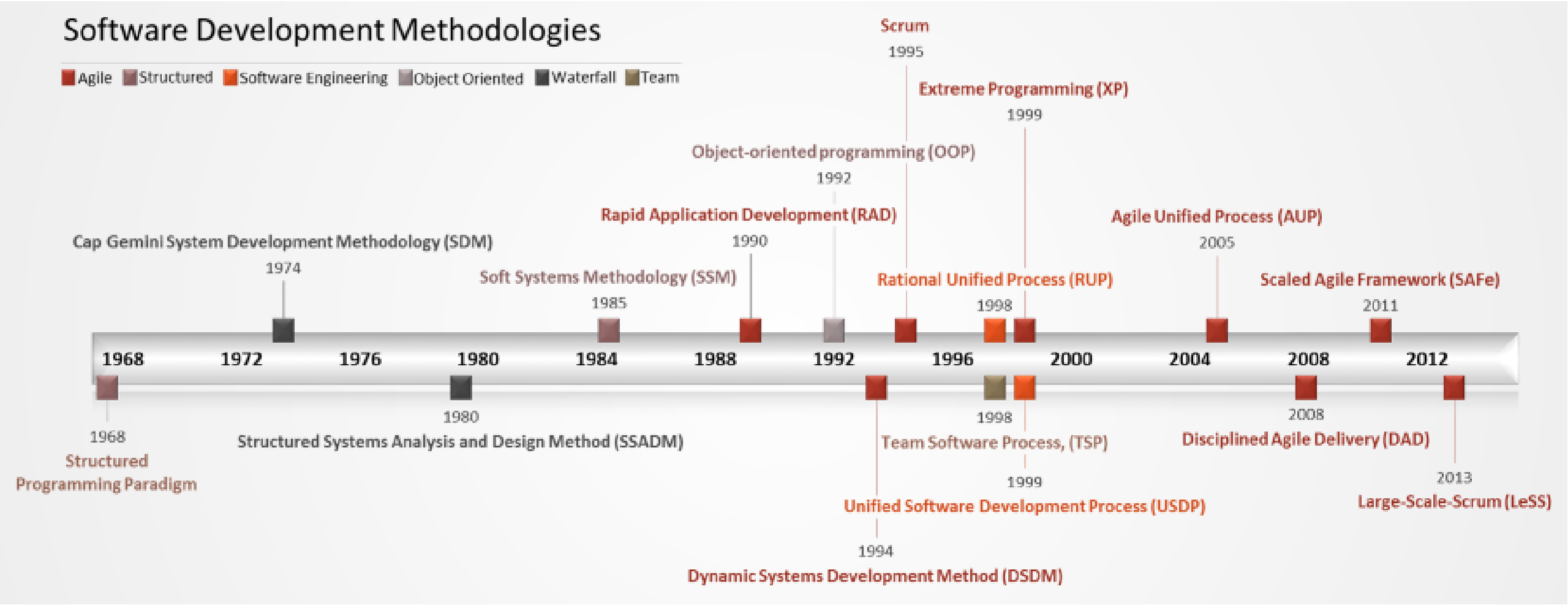
- Software was an abstract concept
- There was no direction for its production, as it did not fit into any of the known engineering's.
- Software made in an “ad-hoc” way, that is, without any associated process
- Coding, in most cases, was done directly without any prior analysis.
    - Flowcharts and textual documentation

SOFTWARE ENGINEERING



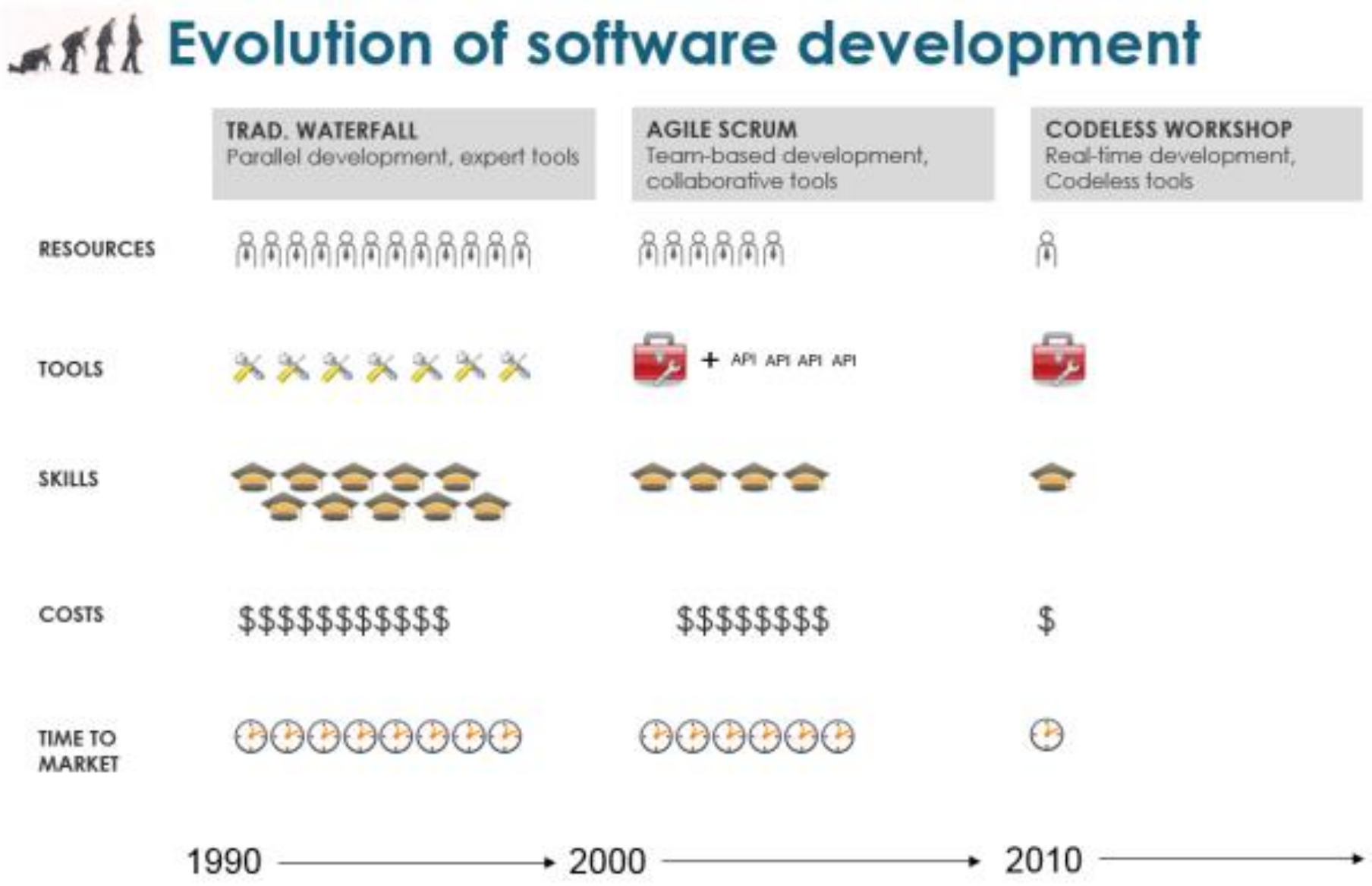
SOFTWARE ENGINEERING

https://www.evolute.be/reviews/safebook.html



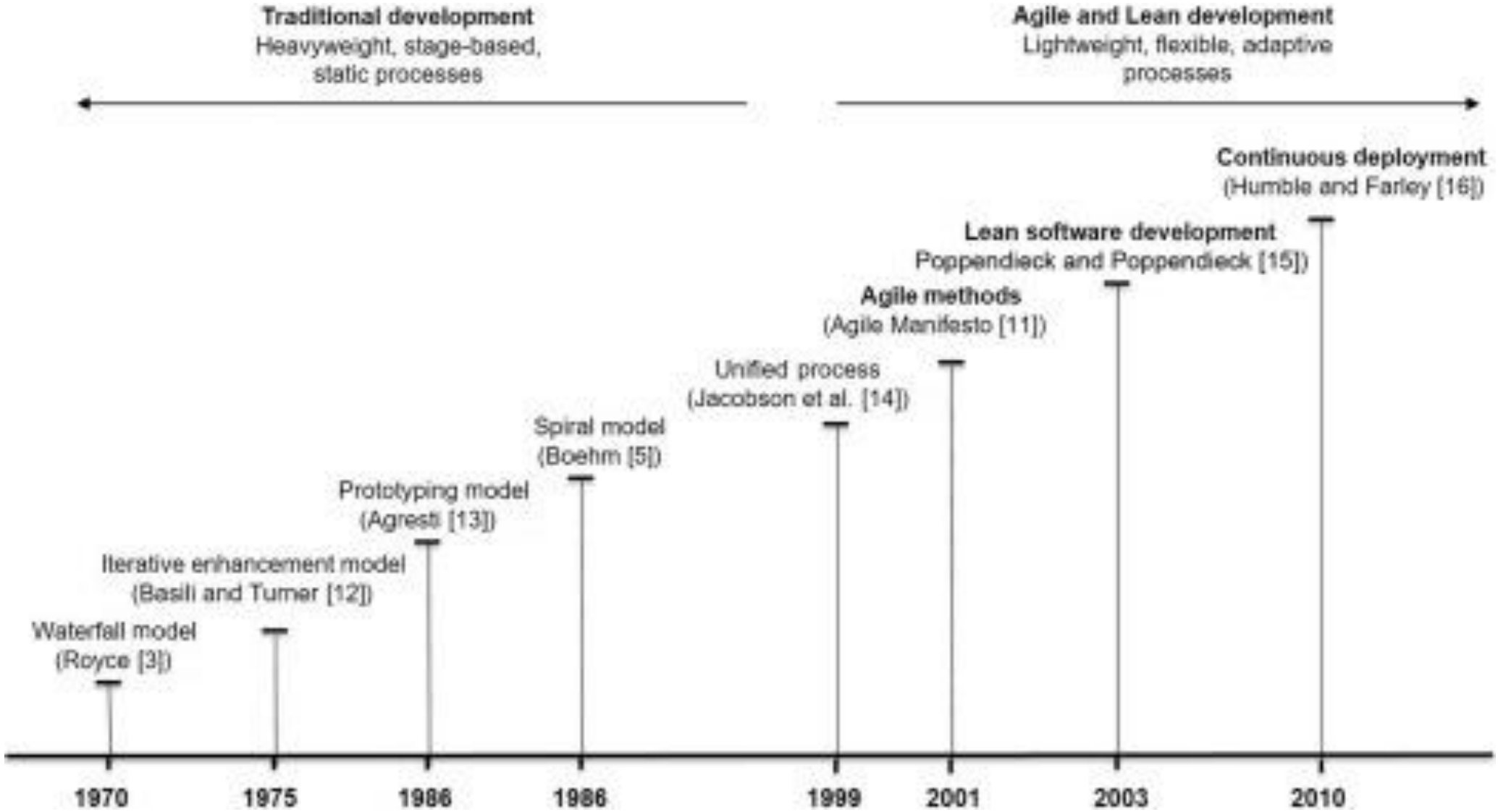
SOFTWARE ENGINEERING

https://www.business2community.com/tech-gadgets/agile-scrums-codeless-workshops-0887767

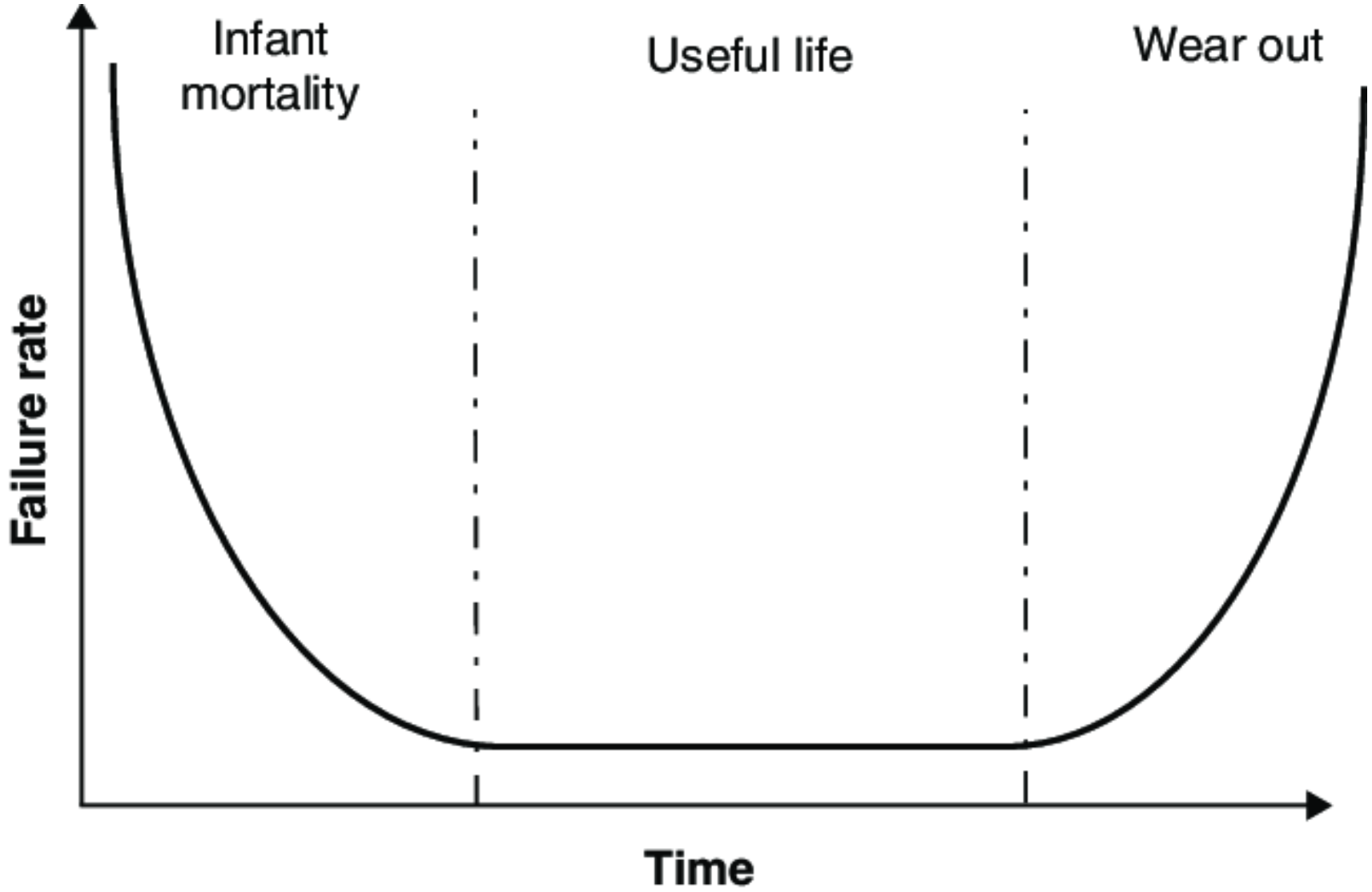


SOFTWARE ENGINEERING

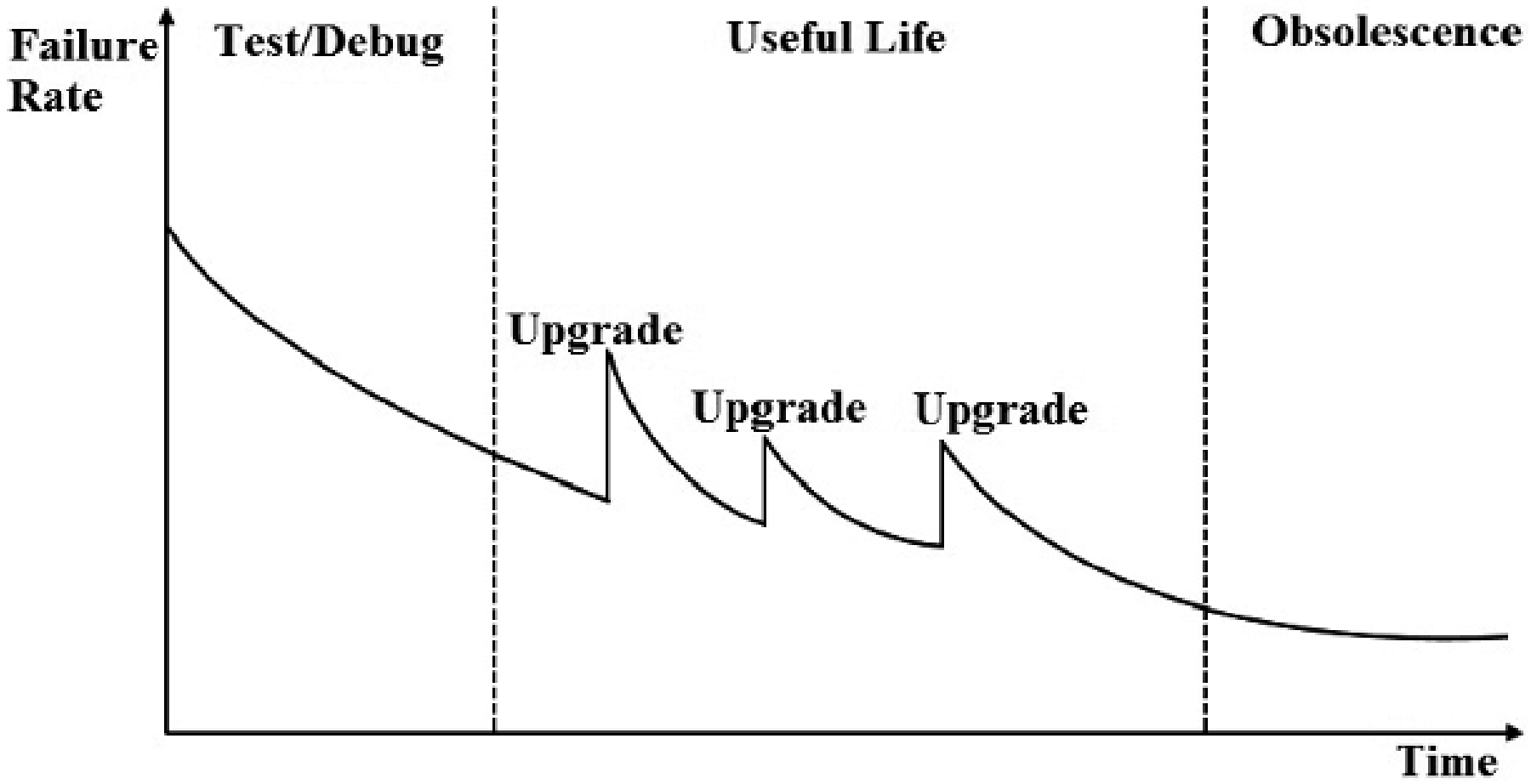
https://www.sciencedirect.com/science/article/abs/pii/S0065245818300299



HARDWARE FAILURE CURVE



SOFTWARE FAILURE CURVE



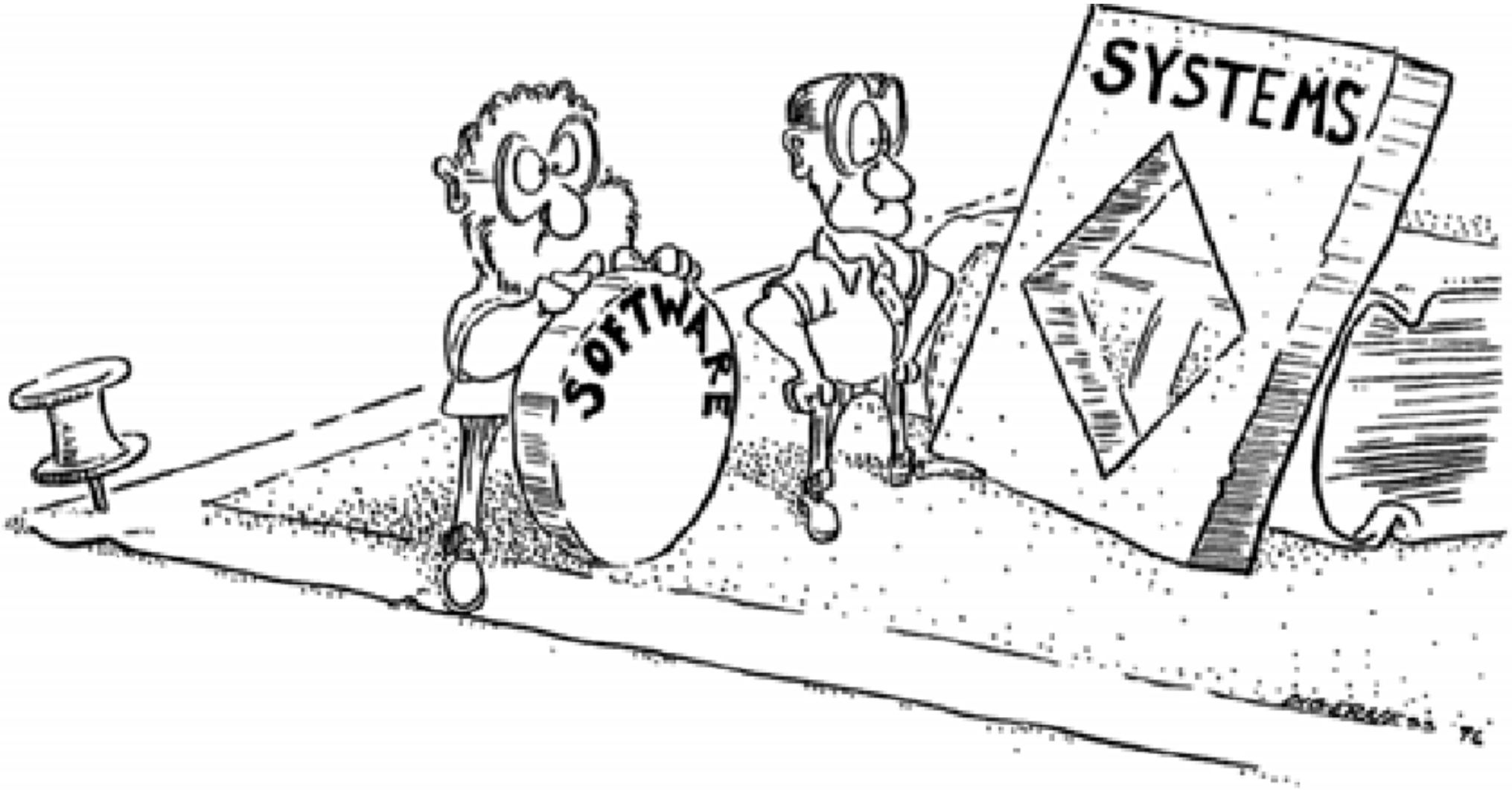
CAUSES OF THE SOFTWARE CRISIS

- Projects that blow the Schedule
- Projects that go over budget
- Low quality
- Dissatisfied customers
- Products difficult to maintain....

And

- Myths...

1968 – THE SOFTWARE CRISIS



1968 – THE SOFTWARE CRISIS

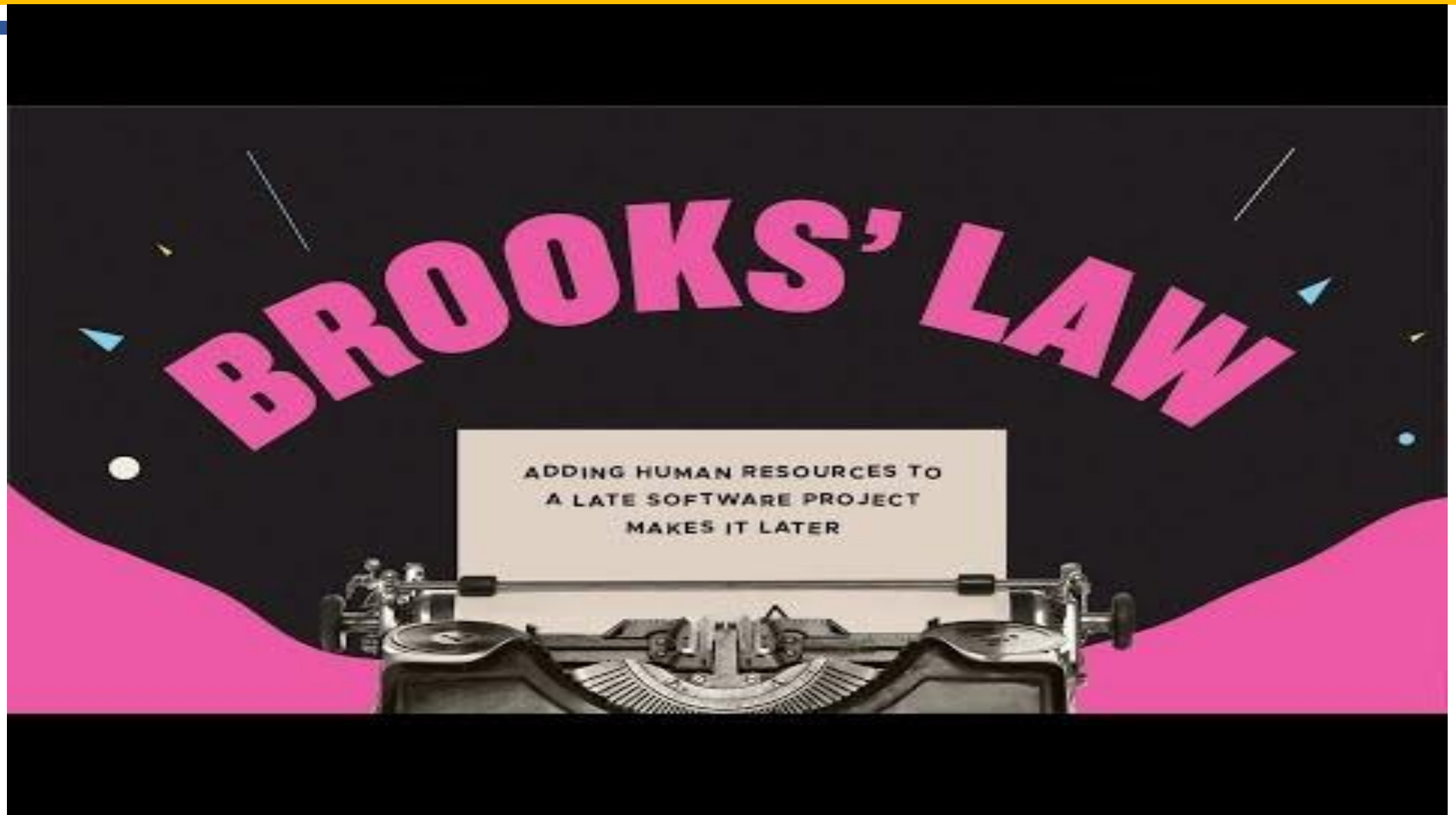
1st NATO (North Atlantic Treaty Organization) software engineering conference

1968 – THE SOFTWARE CRISIS

- Term **software crisis** used for a long time to refer to software problems not keeping up with hardware development
- Recognition of the lack of adequate techniques and processes for the development of quality software
- Dijkstra's statement (+/- 1970): complex and obscure software specifications that make the programmer "the most unfortunate creatures on the face of the earth".

LESSONS FROM "THE MYTHICAL MAN-MONTH"

https://youtu.be/Xsd7rJMmZHg



SOFTWARE ENGINEERING

Frequently asked questions about software engineering

QUESTION	ANSWER
What is software?	Computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market
What are the attributes of good software?	Good software should deliver the required functionality and performance to the user and should be maintainable, dependable and usable
What is software engineering?	Software engineering is an engineering discipline that is concerned with all aspects of software production.
What are the fundamental software engineering activities?	Software specification, software development, software validation and software evolution.
What is the difference between software engineering and computer science?	Computer science focuses on theory and fundamentals. Software engineering is concerned with the practicalities of developing and delivering useful software.
What is the difference between software engineering and system engineering?	System engineering is concerned with all aspects of computer based systems development including hardware, software and process engineering. Software engineering is part of this more general process.

SOFTWARE ENGINEERING

Frequently asked questions about software engineering

QUESTION	ANSWER
What are the key challenges facing software engineering?	Coping with increasing diversity, demands for reduced delivery times and developing trustworthy software
What are the costs of software engineering?	Roughly 60% of software costs are development costs, 40% are testing costs for custom software, evolution costs often exceed development costs
What are the best software engineering techniques and methods?	While all software projects have to be professionally managed and developed, different techniques are appropriate for different types of system For example, games should always be developed using a series of prototypes whereas safety critical control systems require a complete and analyzable specification to be developed You can't, therefore, say that one method is better than another
What differences has the web made to software engineering?	The web has led to the availability of software services and the possibility of developing highly distributed service based systems Web based systems development has led to important advances in programming languages and software reuse.

SOFTWARE PRODUCTS

GENERIC PRODUCTS

- Stand alone systems that are marketed and sold to any customer who wishes to buy them.
  - Examples - PC software such as graphics programs, project management tools; CAD software; software for specific markets such as appointments systems for dentists.

CUSTOMIZED PRODUCTS

- Software that is commissioned by a specific customer to meet their own needs.
  - Examples – embedded control systems, air traffic control software, traffic monitoring systems.

PRODUCT SPECIFICATION

GENERIC PRODUCTS

- The specification of what the software should do is owned by the software developer and decisions on software change are made by the developer.

CUSTOMIZED PRODUCTS

- The specification of what the software should do is owned by the customer for the software and they make decisions on software changes that are required.

SOFTWARE ENGINEERING

Essential attributes of good software

PRODUCT CHARACTERISTIC	DESCRIPTION
Maintainability	Software should be written in such a way so that it can evolve to meet the changing needs of customers This is a critical attribute because software change is an inevitable requirement of a changing business environment
Dependability and security	Software dependability includes a range of characteristics including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure Malicious users should not be able to access or damage the system
Efficiency	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilization, etc.
Acceptability	Software must be acceptable to the type of users for which it is designed This means that it must be understandable, usable and compatible with other systems that they use

SOFTWARE ENGINEERING

SOFTWARE ENGINEERING

Software engineering is an engineering discipline that is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use.

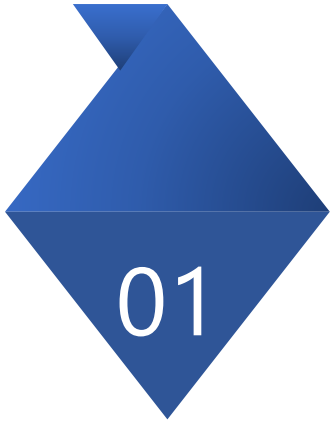
ENGINEERING DISCIPLINE

Using appropriate theories and methods to solve problems bearing in mind organizational and financial constraints.

ALL ASPECTS OF SOFTWARE PRODUCTION

Not just technical process of development. Also project management and the development of tools, methods etc. to support software production.

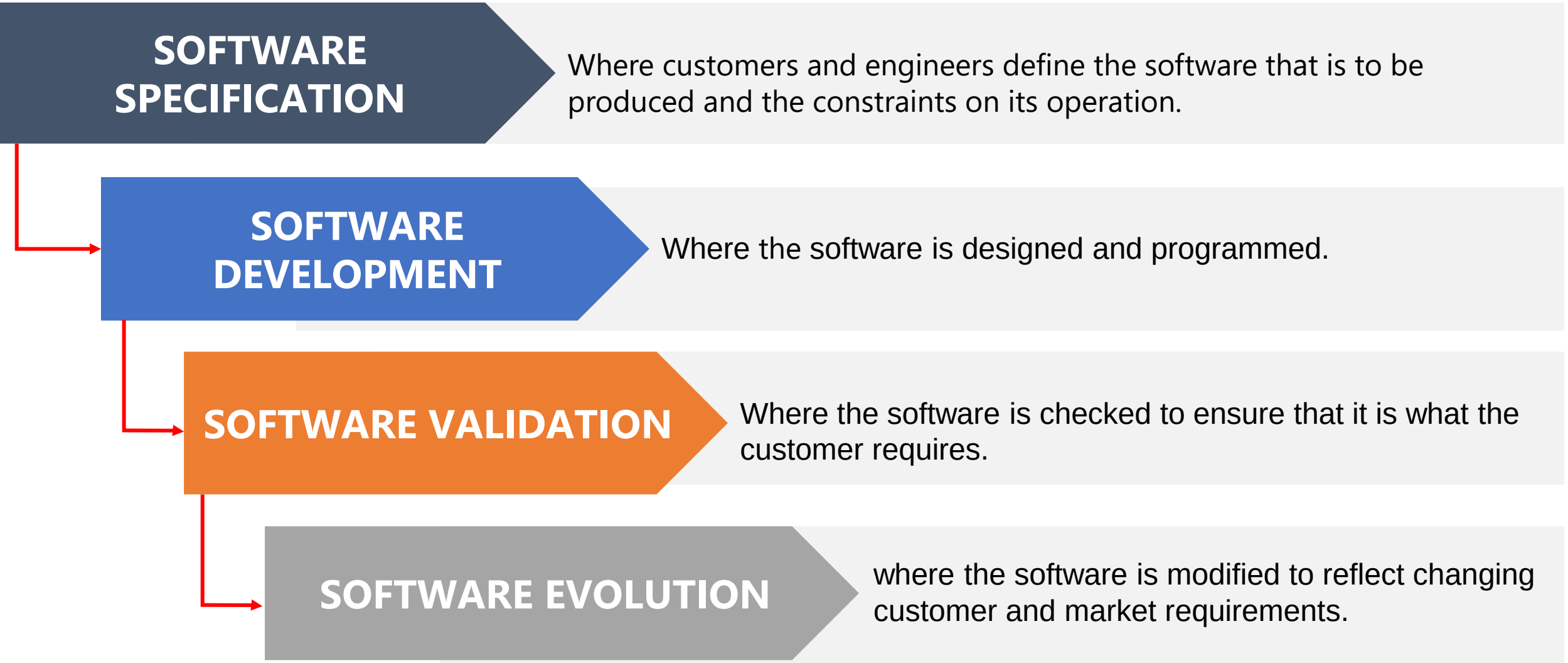
IMPORTANCE OF SOFTWARE ENGINEERING



More and more, individuals and society rely on advanced software systems . We need to be able to produce reliable and trustworthy systems economically and quickly.

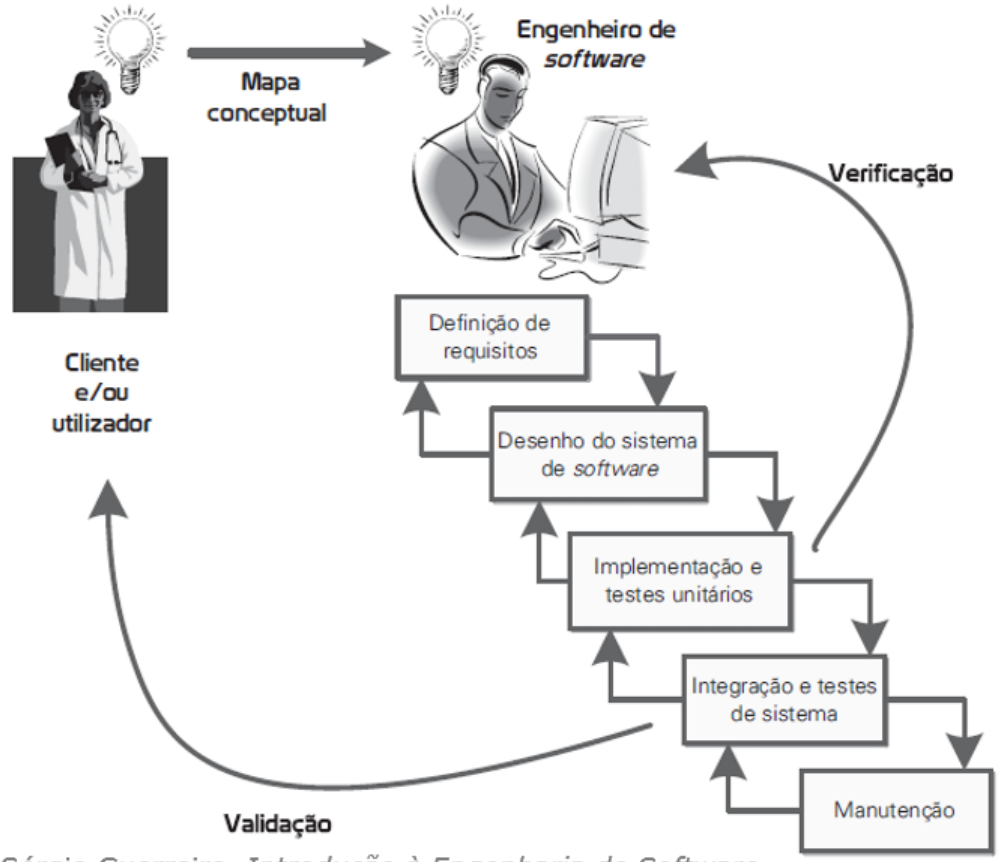
It is usually cheaper, in the long run, to use software engineering methods and techniques for software systems rather than just write the programs as if it was a personal programming project. For most types of system, the majority of costs are the costs of changing the software after it has gone into use.

SOFTWARE PROCESS ACTIVITIES



SOFTWARE PROCESS ACTIVITIES

A COMUNICAÇÃO ENTRE CLIENTES/UTILIZADORES E O ENGENHEIRO DE *SOFTWARE* NO CONTEXTO DO PROCESSO DE DESENVOLVIMENTO



Fonte: Sérgio Guerreiro, *Introdução à Engenharia de Software*, FCA – Editora de Informática

General issues that affect software

HETEROGENEITY

- Increasingly, systems are required to operate as distributed systems across networks that include different types of computer and mobile devices.

BUSINESS AND SOCIAL CHANGE

- Business and society are changing incredibly quickly as emerging economies develop and new technologies become available. They need to be able to change their existing software and to rapidly develop new software.

General issues that affect software

SECURITY AND TRUST

- As software is intertwined with all aspects of our lives, it is essential that we can trust that software.

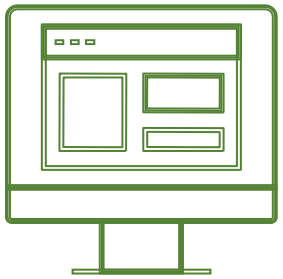
SCALE

- Software has to be developed across a very wide range of scales, from very small embedded systems in portable or wearable devices through to Internet scale, cloud based systems that serve a global community.

Software engineering diversity

- **There are many different types of software systems** and **there is no universal set of software techniques** that is applicable to all of these.
- The **software engineering methods and tools** used depend on the **type of application being developed** , the requirements of the customer and the background of the development team.

APPLICATIONS TYPES

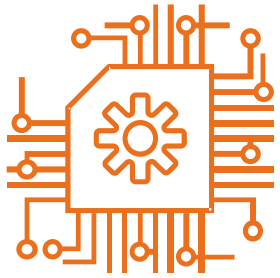


STAND ALONE APPLICATIONS

- These are application systems that run on a local computer, such as a PC. They include all necessary functionality and do not need to be connected to a network.

INTERACTIVE TRANSACTION BASED APPLICATIONS

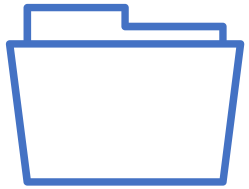
- Applications that execute on a remote computer and are accessed by users from their own PCs or terminals. These include web applications such as e commerce applications.



EMBEDDED CONTROL SYSTEMS

- These are software control systems that control and manage hardware devices. Numerically, there are probably more embedded systems than any other type of system.

APPLICATIONS TYPES

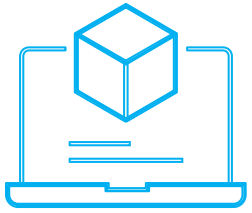


BATCH PROCESSING SYSTEMS

- These are business systems that are designed to process data in large batches. They process large numbers of individual inputs to create corresponding outputs.

ENTERTAINMENT SYSTEMS

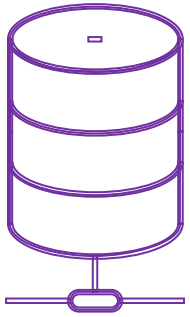
- These are systems that are primarily for personal use and which are intended to entertain the user.



SYSTEMS FOR MODELLING AND SIMULATION

- These are systems that are developed by scientists and engineers to model physical processes or situations, which include many, separate, interacting objects..

APPLICATIONS TYPES



DATA COLLECTION SYSTEMS

- These are systems that collect data from their environment using a set of sensors and send that data to other systems for processing.

SYSTEMS OF SYSTEMS

- These are systems that are composed of a number of other software systems.

Software engineering fundamentals

Some fundamental principles apply to all types of software system, irrespective of the development techniques used:

- **Systems should be developed using a managed and understood development process**. Of course, different processes are used for different types of software.
- **Dependability and performance** are important for all types of system.
- **Understanding and managing the software specification and requirements** (what the software should do) are important.
- Where appropriate, you should **reuse software** that has already been developed rather than write new software.

Internet software engineering

- The Web is now a platform for running application and organizations are increasingly developing web based systems rather than local systems.
- Web services allow application functionality to be accessed over the web.
- Cloud computing is an approach to the provision of computer services where applications run remotely on the ‘cloud’.
  - Users do not buy software buy pay according to use.

Web-based software engineering

- **Web based systems** are complex distributed systems but the fundamental principles of software engineering discussed previously are as applicable to them as they are to any other types of system.
- The fundamental ideas of software engineering apply to web based software **in the same way** that they apply to other types of software system.

SOFTWARE ENGINEERING

Web software engineering

SOFTWARE REUSE

- Software reuse is the dominant approach for constructing web based systems. When building these systems, you think about how you can assemble them from pre existing software components and systems.

INCREMENTAL AND AGILE DEVELOPMENT

- Web based systems should be developed and delivered incrementally. It is now generally recognized that it is impractical to specify all the requirements for such systems in advance.

SOFTWARE ENGINEERING

Web software engineering

SERVICE ORIENTED SYSTEMS

- Software may be implemented using service oriented software engineering, where the software components are stand-alone web services.

RICH INTERFACES

- Interface development technologies such as AJAX and HTML5 have emerged that support the creation of rich interfaces within a web browser.

Web software engineering

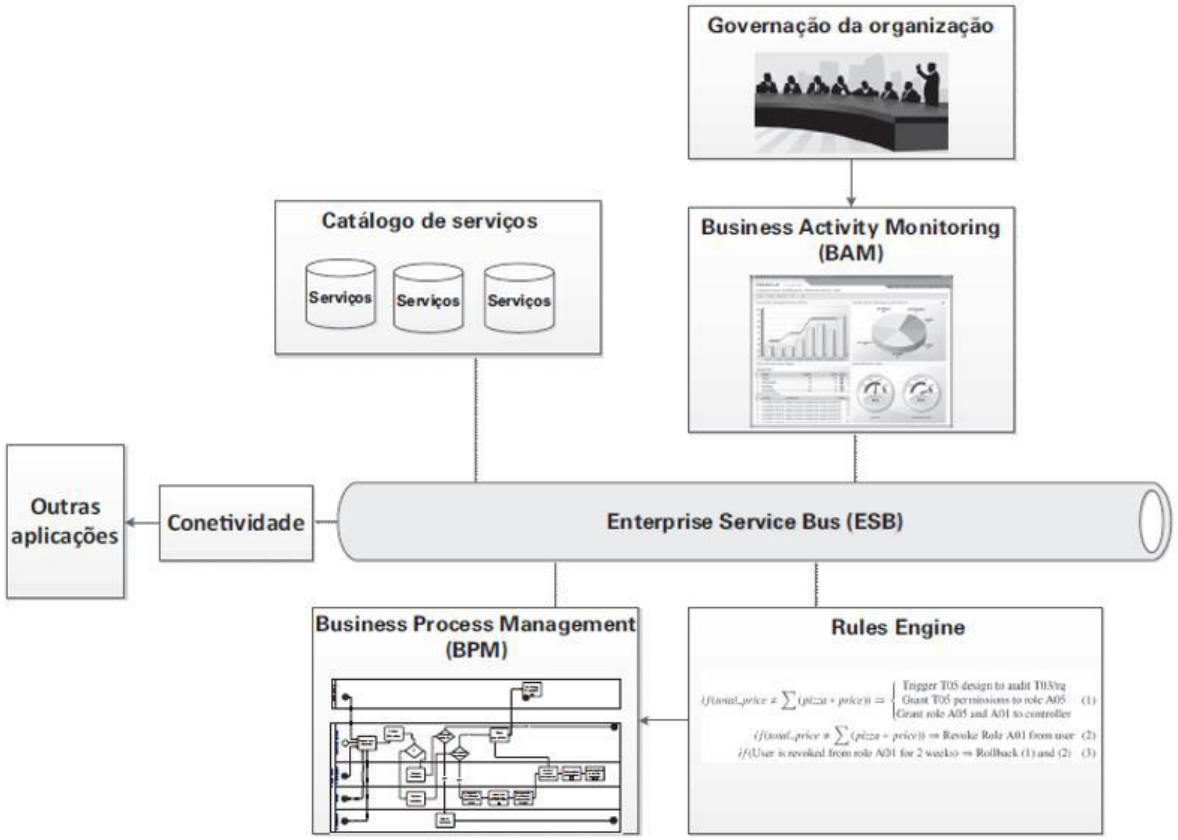


FIGURA 1.7 – EXEMPLO DE ARQUITETURA DE *SOFTWARE* ORIENTADA A SERVIÇOS

Fonte: Sérgio Guerreiro, *Introdução à Engenharia de Software*, FCA – Editora de Informática

SOFTWARE ENGINEERING ETHICS

SOFTWARE ENGINEERING

software engineering ethics

Software engineering involves wider responsibilities than simply the application of technical skills.

Software engineers must behave in an honest and ethically responsible way if they are to be respected as professionals.

Ethical behavior is more than simply upholding the law but involves following a **set of principles that are morally correct**.

Issues of professional responsibility

Confidentiality

- Engineers should normally respect the confidentiality of their employers or clients irrespective of whether or not a formal confidentiality agreement has been signed.

Competence

- Engineers should not misrepresent their level of competence. They should not knowingly accept work which is outwith their competence.

SOFTWARE ENGINEERING ETHICS

Issues of professional responsibility

Intellectual property rights

- Engineers should be aware of local laws governing the use of intellectual property such as patents, copyright, etc. They should be careful to ensure that the intellectual property of employers and clients is protected.

Computer misuse

- Software engineers should not use their technical skills to misuse other people's computers. Computer misuse ranges from relatively trivial (game playing on an employer's machine, say) to extremely serious (dissemination of viruses).

ACM/IEEE Code of Ethics

- The professional societies in the US have cooperated to produce a code of ethical practice.
- Members of these organizations sign up to the code of practice when they join.
- The Code contains eight Principles related to the behavior of and decisions made by professional software engineers, including practitioners, educators, managers, supervisors and policy makers, as well as trainees and students of the profession.

Rationale for the code of ethics

- *Computers have a central and growing role in commerce, industry, government, medicine, education, entertainment and society at large. Software engineers are those who contribute by direct participation or by teaching, to the analysis, specification, design, development, certification, maintenance and testing of software systems.*
- *Because of their roles in developing software systems, software engineers have significant opportunities to do good or cause harm, to enable others to do good or cause harm, or to influence others to do good or cause harm. To ensure, as much as possible, that their efforts will be used for good, software engineers must commit themselves to making software engineering a beneficial and respected profession.*

ACM/IEEE Code of Ethics

Software Engineering Code of Ethics and Professional Practice

ACM/IEEE CS Joint Task Force on Software Engineering Ethics and Professional Practices

PREAMBLE

The short version of the code summarizes aspirations at a high level of the abstraction; the clauses that are included in the full version give examples and details of how these aspirations change the way we act as software engineering professionals. Without the aspirations, the details can become legalistic and tedious; without the details, the aspirations can become high sounding but empty; together, the aspirations and the details form a cohesive code. Software engineers shall commit themselves to making the analysis, specification, design, development, testing and maintenance of software a beneficial and respected profession. In accordance with their commitment to the health, safety and welfare of the public, software engineers shall adhere to the following Eight Principles:

ETHICAL PRINCIPLES

PUBLIC
Software engineers shall act consistently with the public interest.

CLIENT AND EMPLOYER
Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.

PRODUCT
Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance..

JUDGMENT
Software engineers shall maintain integrity and independence in their professional judgment..

1

2

3

4

5

6

7

8

MANAGEMENT
Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance..

PROFESSION
Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.

COLLEAGUES
Software engineers shall be fair to and supportive of their colleagues.

SELF
Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

ETHICAL DILEMAS

- Disagreement in principle with the policies of senior management
- Your employer acts in an unethical way and releases a safety critical system without finishing the testing of the system.
- Participation in the development of military weapons systems or nuclear systems.

CASE STUDIES

A personal insulin pump

An embedded system in an insulin pump used by diabetics to maintain blood glucose control.

CS01

Airbus 340 flight control system

Architecture of the Airbus 340 flight control system, a safety critical system that implements the fly-by-wire flight system on the Airbus.

CS02

Ariane 5 launch accident

CS03

Describes the accident that occurred on the initial launch of the Ariane 5 rocket, a launcher developed by the European Space Agency.

CS04

iLearn: a digital learning environment

A system to support learning in schools

CASE STUDIES

INSULIN PUMP CONTROL SYSTEM

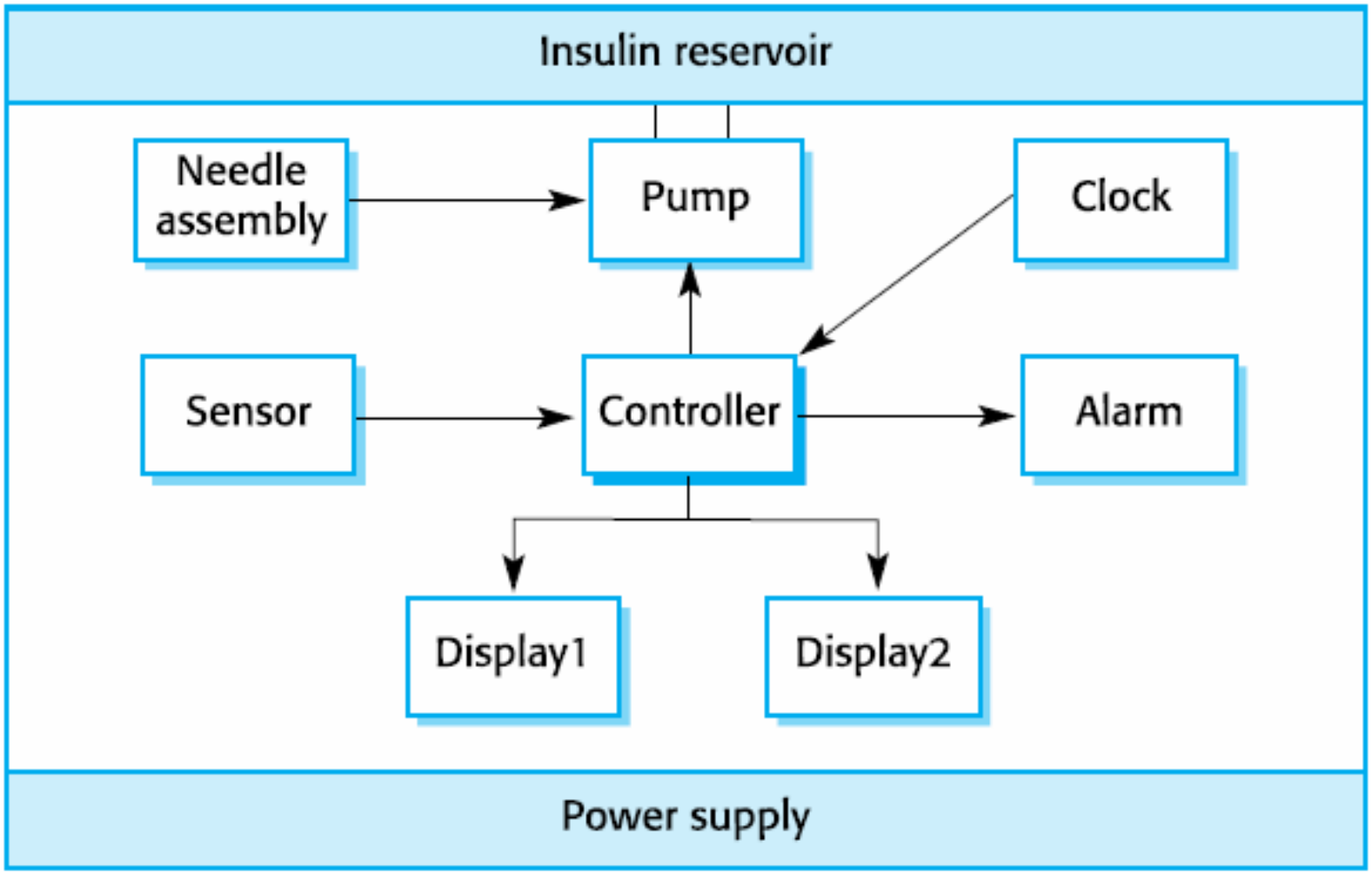
CS01

INSULIN PUMP CONTROL SYSTEM

- Collects data from a blood sugar sensor and calculates the amount of insulin required to be injected.
- Calculation based on the rate of change of blood sugar levels.
- Sends signals to a micro pump to deliver the correct dose of insulin.
- Safety critical system as low blood sugars can lead to brain malfunctioning, coma and death; high blood sugar levels have long term consequences such as eye and kidney damage.

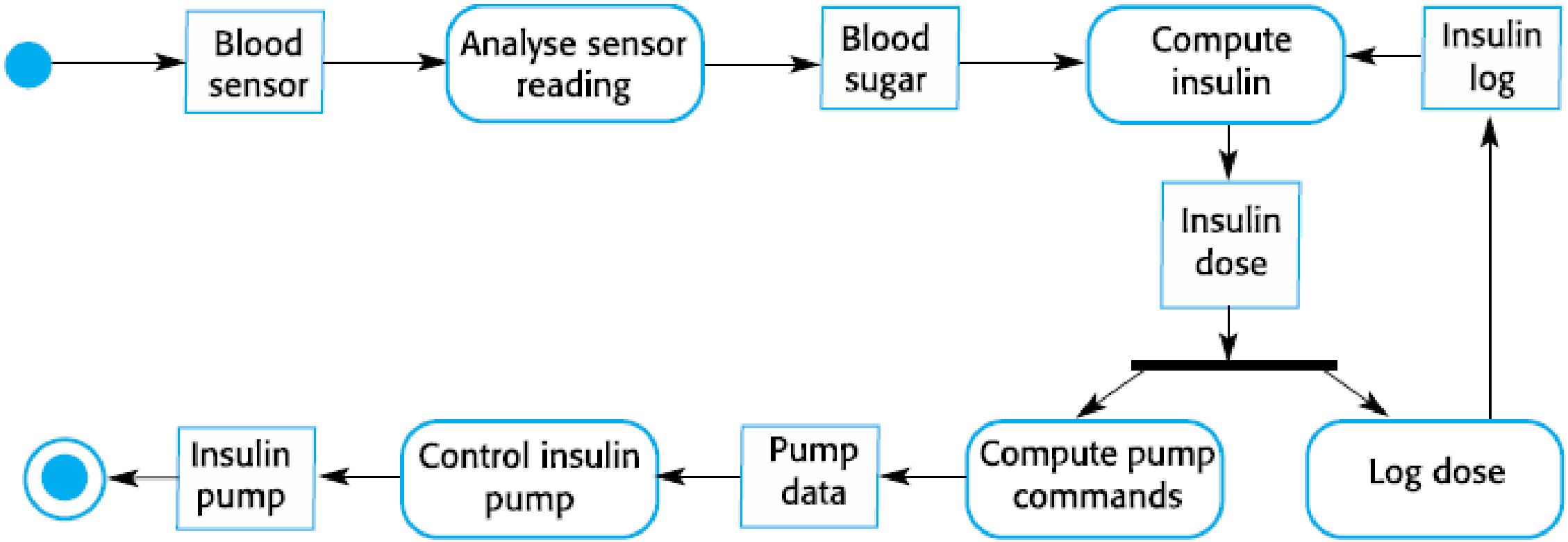
CS01

INSULIN PUMP HARDWARE ARCHITECTURE



CS01

ACTIVITY MODEL OF THE INSULIN PUMP



CS01

ESSENTIAL HIGH LEVEL REQUIREMENTS

The system shall be available to deliver insulin when required.

The system shall perform reliably and deliver the correct amount of insulin to counteract the current level of blood sugar.

The system must therefore be designed and implemented to ensure that the system always meets these requirements.

CASE STUDIES

AIRBUS 340 FLIGHT CONTROL SYSTEM

CS02

AIRBUS 340 FLIGHT CONTROL SYSTEM

Explains the organization of the safety-critical flight control system on the Airbus 330/340 and how redundancy and diversity is used in that system

CASE STUDIES

ARIANE 5 LAUNCH ACCIDENT

CS03

ARIANE 5 LAUNCH ACCIDENT

A video of the take-off and explosion after 37 seconds

https://youtu.be/ZY2J4p86oPw

CS03

ARIANE 5 LAUNCH ACCIDENT

Explains why a software failure on the first launch of the Ariane 5 rocket was responsible for the failure and complete destruction of the rocket and its payload

https://www.youtube.com/watch?v=W3YJeoYgozw

CASE STUDIES

ILEARN: A DIGITAL LEARNING ENVIRONMENT

CS04

iLEARN: A DIGITAL LEARNING ENVIRONMENT

- A digital learning environment is a framework in which a set of general purpose and specially designed tools for learning may be embedded plus a set of applications that are geared to the needs of the learners using the system.

- The tools included in each version of the environment are chosen by teachers and learners to suit their specific needs.
  - These can be general applications such as spreadsheets , learning management applications such as a Virtual Learning Environment (VLE) to manage homework submission and assessment, games and simulations.

CS04

SERVICE ORIENTED SYSTEMS

The system is a service oriented system with all system components considered to be a replaceable service.

This allows the system to be updated incrementally as new services become available.

It also makes it possible to rapidly configure the system to create versions of the environment for different groups such as very young children who cannot read, senior students, etc.

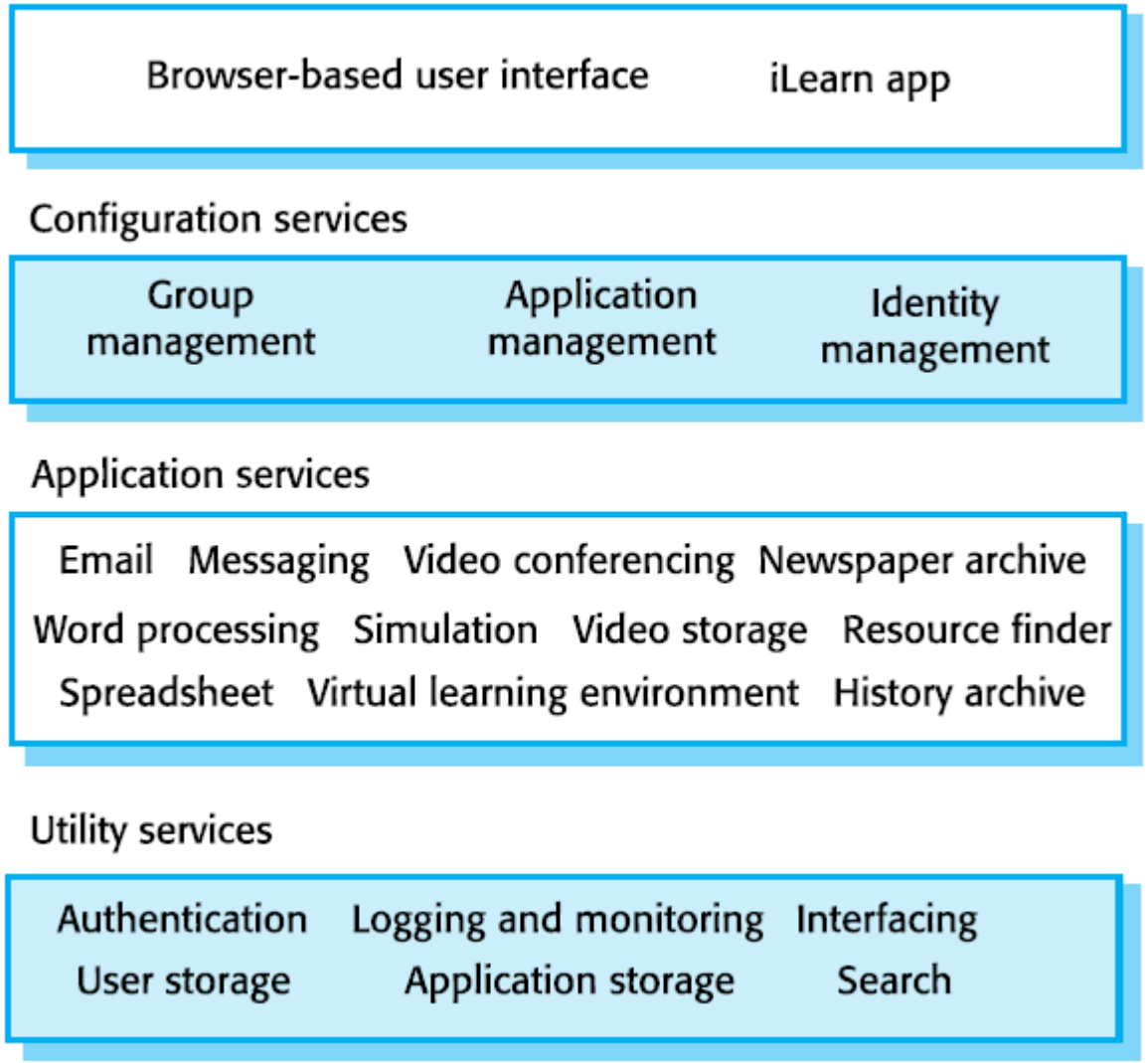
CS04

ILEARN SERVICES

- *Utility services* that provide basic application-independent functionality and which may be used by other services in the system.
- *Application services* that provide specific applications such as email, conferencing, photo sharing etc. and access to specific educational content such as scientific films or historical resources.
- *Configuration services* that are used to adapt the environment with a specific set of application services and do define how services are shared between students, teachers and their parents.

CS04

ILEARN ARCHITECTURE



CS04

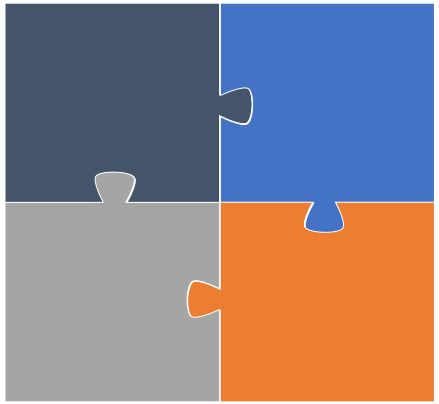
ILEARN SERVICE INTEGRATION

- Integrated services are services which offer an API (application programming interface) and which can be accessed by other services through that API. Direct service to service communication is therefore possible.
- Independent services are services which are simply accessed through a browser interface and which operate independently of other services. Information can only be shared with other services through explicit user actions such as copy and paste; re authentication may be required for each independent service.

KEY POINTS

Essential software product attributes are maintainability, dependability and security, efficiency and acceptability.

Software engineering is an engineering discipline that is concerned with all aspects of software production.



The fundamental notions of software engineering are universally applicable to all types of system development.

The high level activities of specification, development, validation and evolution are part of all software processes.

KEY POINTS

The fundamental ideas of software engineering are applicable to all types of software system.

There are many different types of system and each requires appropriate software engineering tools and techniques for their development.

Software engineers have responsibilities to the engineering profession and society. They should not simply be concerned with technical issues.

Professional societies publish codes of conduct which set out the standards of behaviour expected of their members.