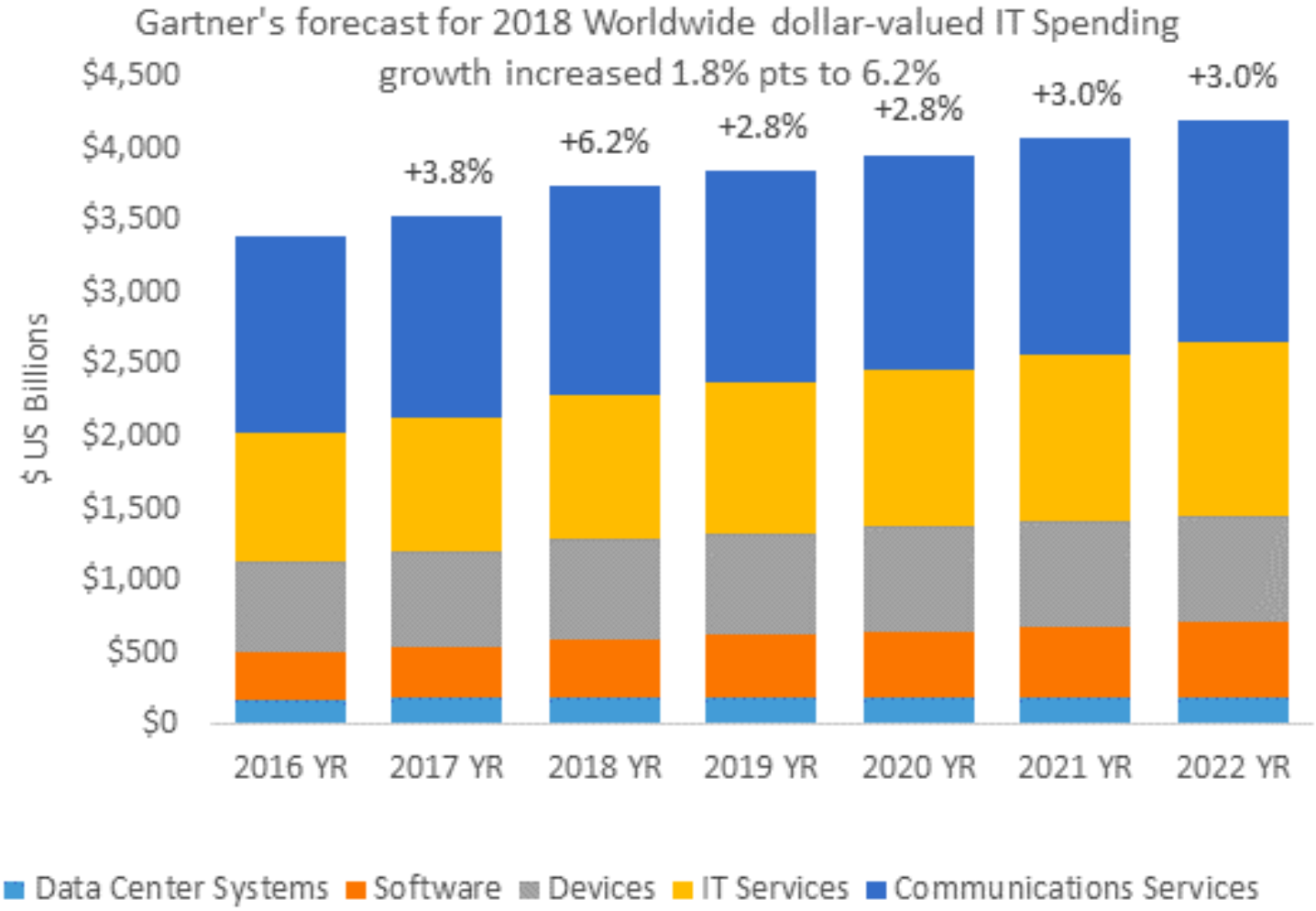
Overview of Software Engineering

SOftEng
http://softeng.polito.it

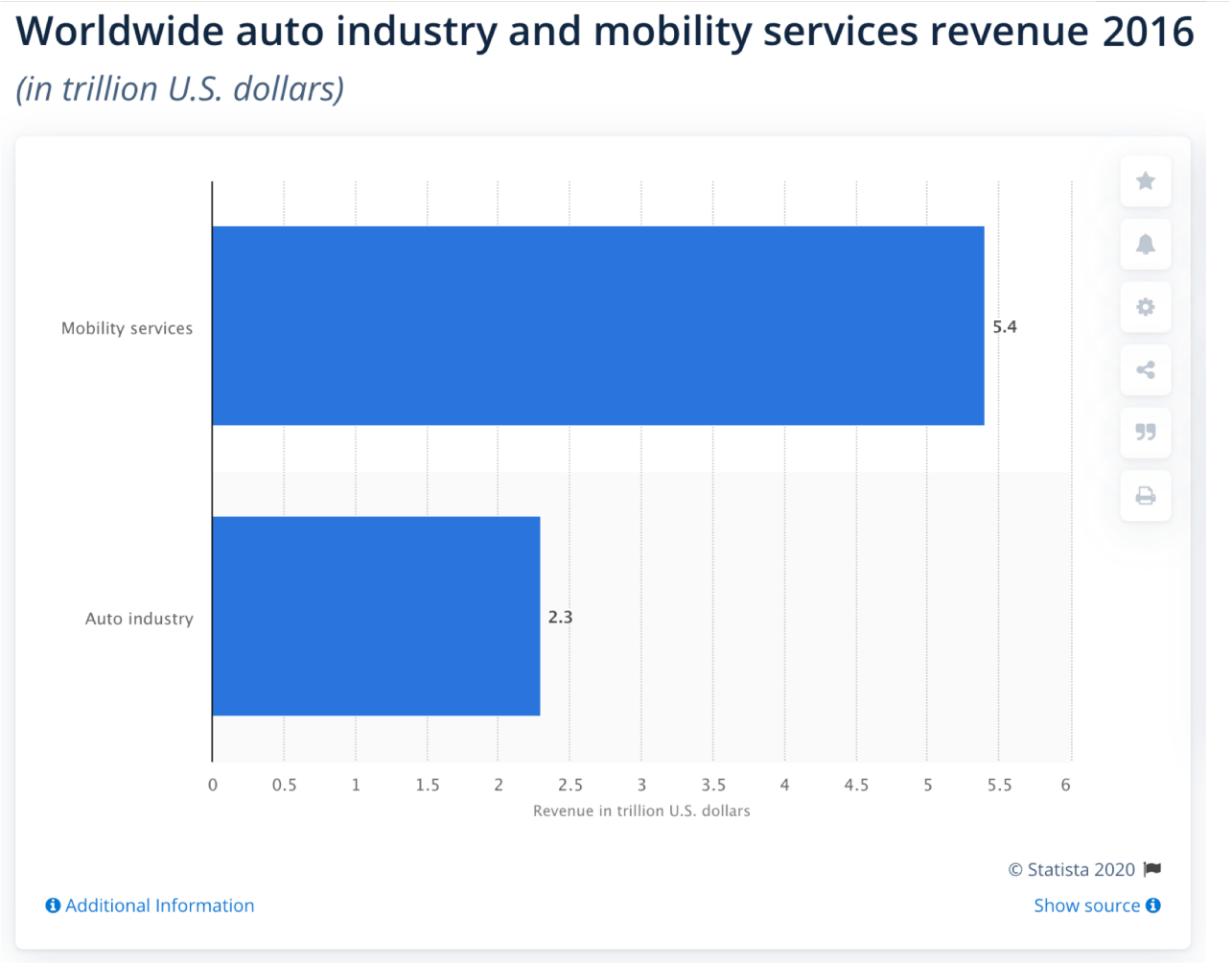
Motivation

SOftEng
http://softeng.polito.it

ICT Market (Global)



ICT Market (Global)

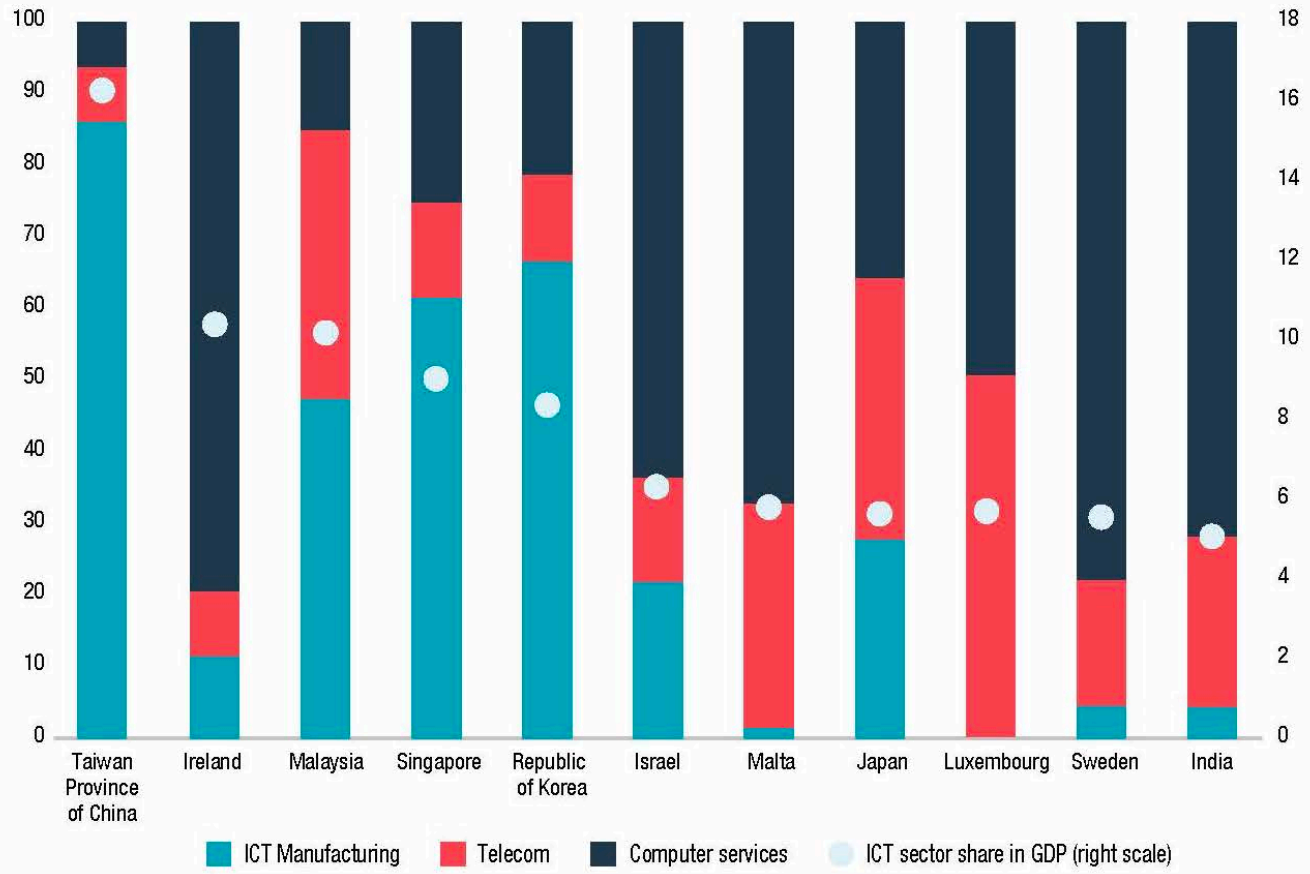


Software and the economy

- The economies of ALL developed nations are dependent on software
- More and more systems are software controlled
- Expenditure on software represents a significant fraction of GNP in all developed countries
- Software engineering: how to develop software

ICT Market (World)

Figure III.3. Share of the ICT sector's value added in GDP, and its distribution by subsector: Top 10 economies, 2017
(Per cent)

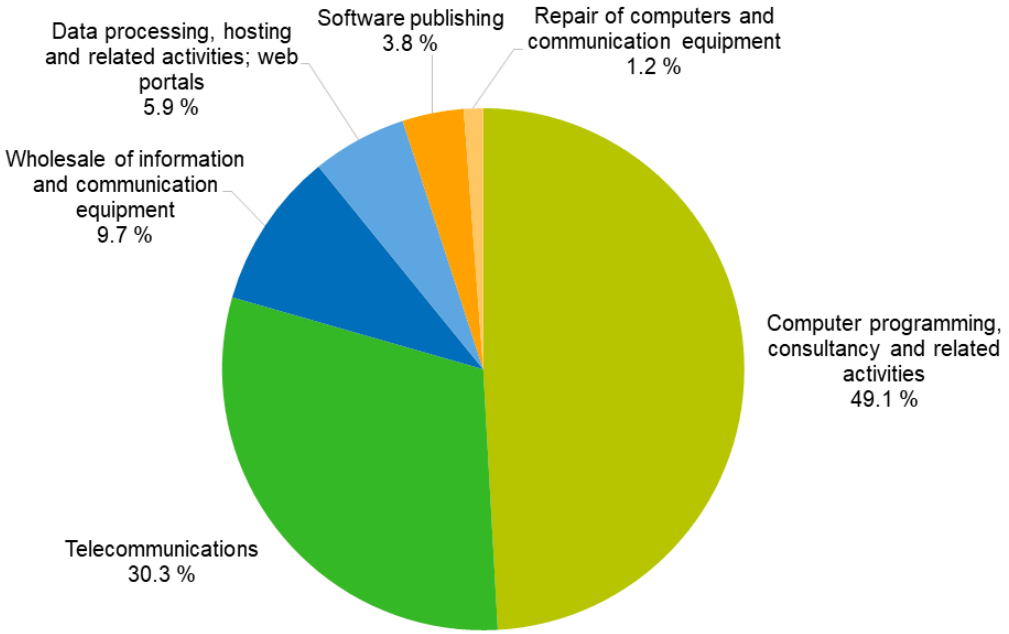


Source: UNCTAD, based on international and national sources (see annex to this chapter).
Note: Data for Ireland refer to 2014, China and India to 2015, and Israel, Japan and Malaysia to 2016.

SOftEng
http://softeng.polito.it

ICT Market (EU)

Distribution of value added within ICT services, EU, 2017
(%)



Note: EU aggregate excluding Ireland, Luxembourg and Malta; including Estonia and Slovakia, 2015.
Data for Greece: provisional.
Due to rounding, individual items may not sum to the total.
Source: online data codes (sbs_na_dt_r2 and sbs_na_1a_se_r2)

eurostat

ICT Market (Italy)

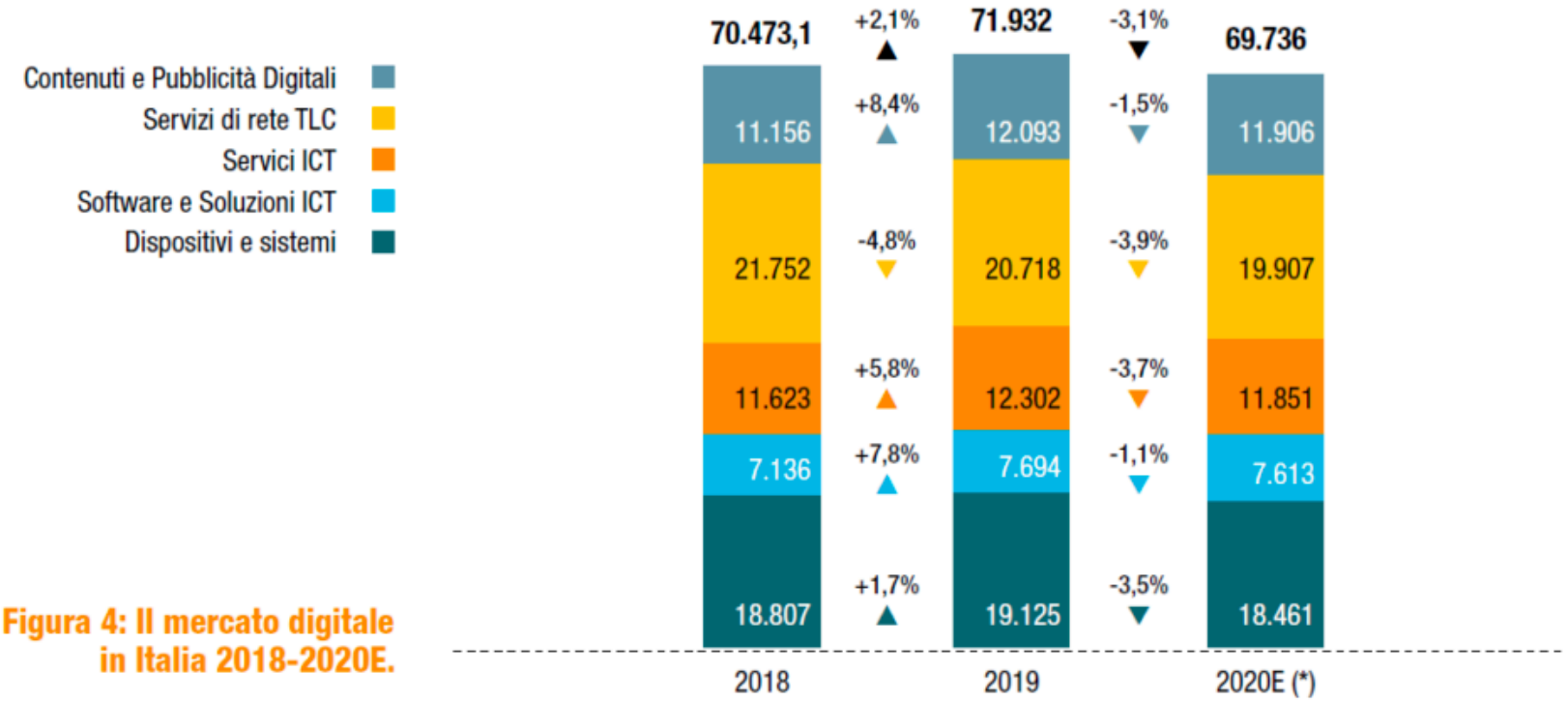


Figura 4: Il mercato digitale in Italia 2018-2020E.

Valori in milioni di euro, variazioni %

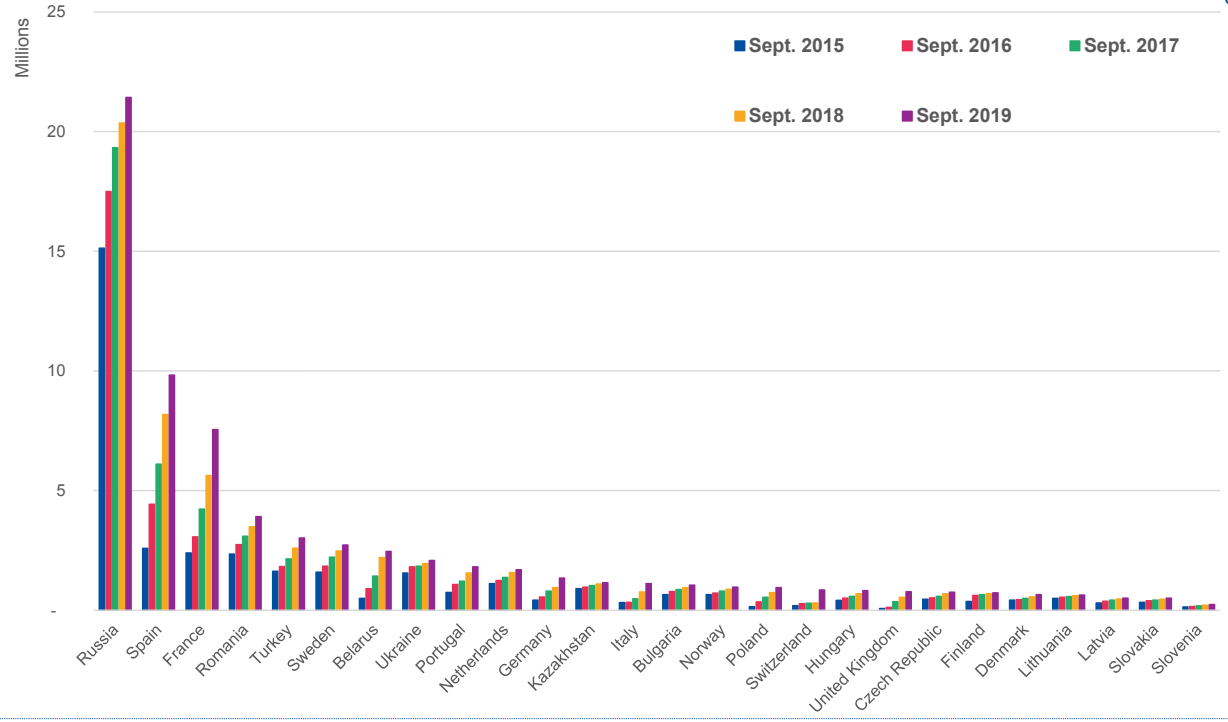
Fonte: NetConsulting cube, maggio 2020

(*) Previsioni provvisorie effettuate sulla base di un valore medio in una forchetta compresa tra -2,5% e -5%, sulla base di una ripresa delle attività produttive entro maggio 2020

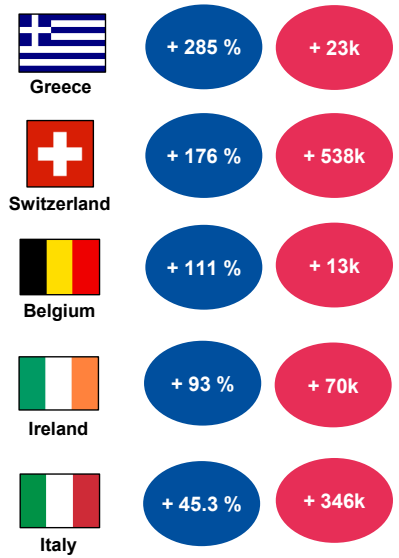
ICT Diffusion (EU)

General Ranking: FTTH/B Subscribers

European ranking in terms of FTTH/B Subscriptions (million)
Data comparison between Sept. 2015 and Sept. 2019



5 fastest growing markets in Europe
Data from Sept. 2018 to Sept. 2019 (in terms of FTTH/B Subscribers)



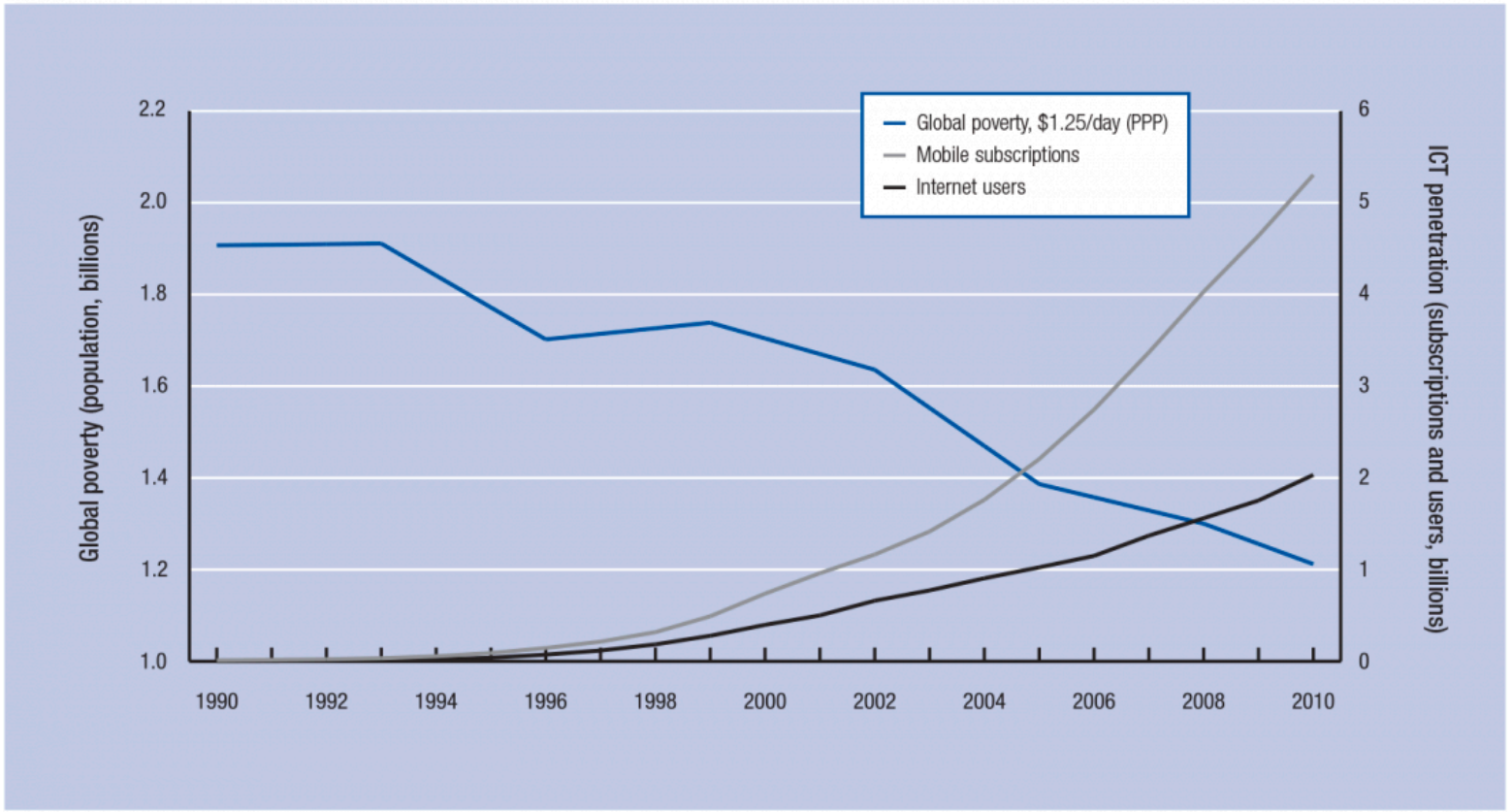
IDATE DIGIWORLD Enlighten your digital future!

Source: IDATE for FTTH Council EUROPE

SoftEng http://softeng.polito.it

ICT Diffusion (Global)

Figure 1: Falling global absolute poverty and rising ICT penetration



Sources: World Bank *PovCal* database (1990, 1993, 1996, 1999, 2002, 2005, 2008, 2010); authors' calculations and interpolation, ITU *World Telecommunication/ICT Indicators* database June 2013.

Innovation and development

- Evidence of correlation between ICT diffusion and wealth
  - Positive correlation IT usage and per capita GDP
  - Positive correlation productivity increase and ICT usage

Definitions

SOftEng
http://softeng.polito.it

Software ≠ Program

- Software is a collection of computer programs, procedures, rules, documentation and data
  - Software development is more than merely the development of programs
  - Software incorporates documents describing various views for various stakeholders (e.g. users, developers)
- For a given problem, software is approximately 10 times more expensive to produce than a simple program
  - Average: 10 to 50 LoC per person day
  - About 7 LoC in critical systems

Software – types

- Stand alone
  - Word processor, …
- Embedded
  - ABS, digital camera, mobile phone, …
- Process support
  - Production process (things): industrial automation
  - Business process (information): management automation

Software – criticality

- Safety critical
  - Aerospace, military, medical, ...
- Mission critical
  - Banking, logistics, industrial production, ...
- Other
  - Games, ...

Software diffusion

- As a result, computers and software became commonplace:
  - In embedded systems (such as cameras, cars, lifts)
  - In industrial systems (such as factories dedicated to manufacturing or continuous processes)
  - In desktop computers (games, personal productivity tools)
  - In businesses (banks, retail stores, etc.)

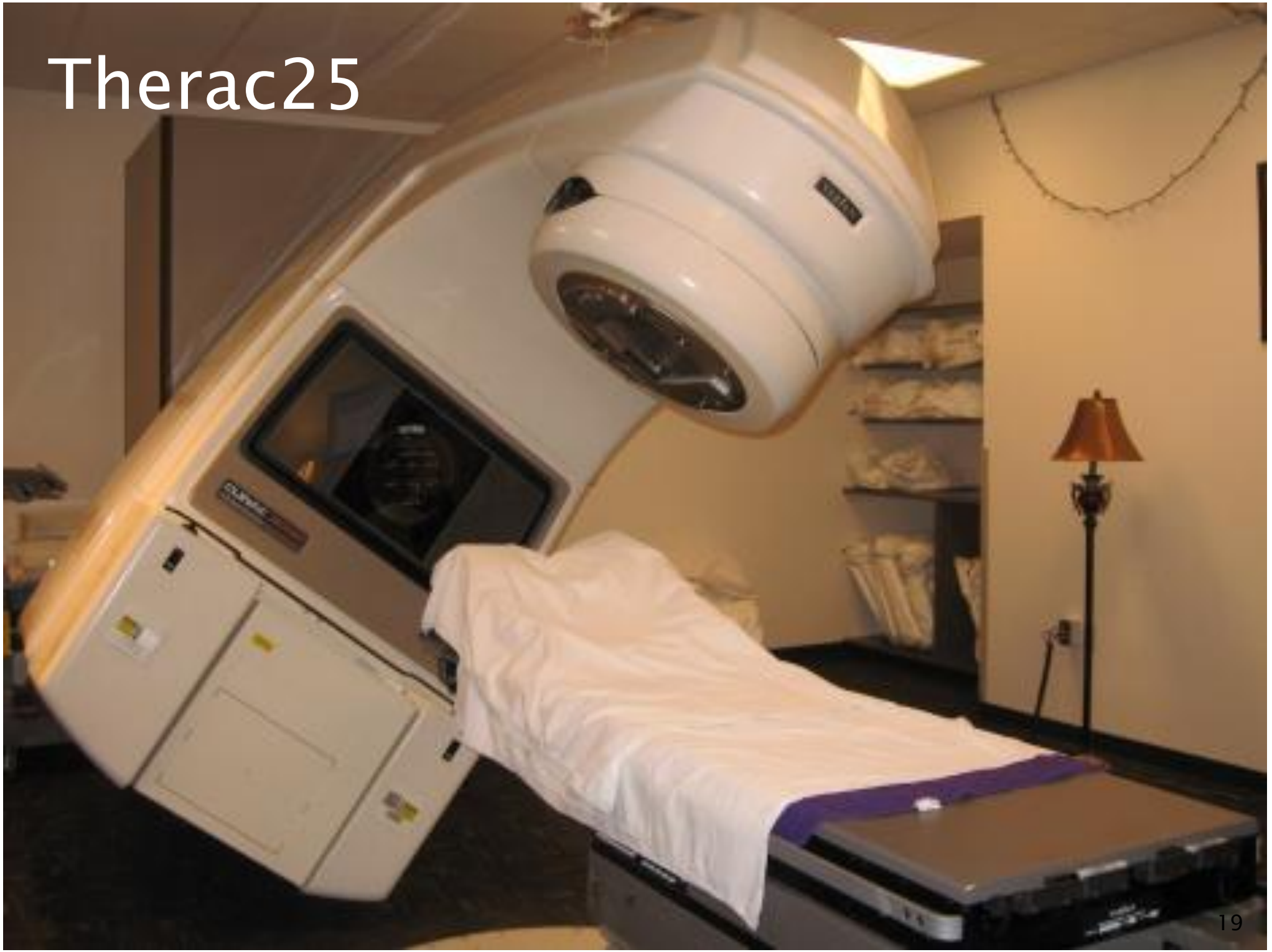
Software diffusion

- The diffusion of software grows constantly in all domains, and also its size (and supported functionalities) grows
- Printer driver
  - 10KLOC 1990, 2MLOC today
- Automotive
  - 20-30 ECU in a low level car
  - 25-40% of cost for car design from electronics and software
- Airplanes
  - Cost of avionics is >50% of cost of airplane

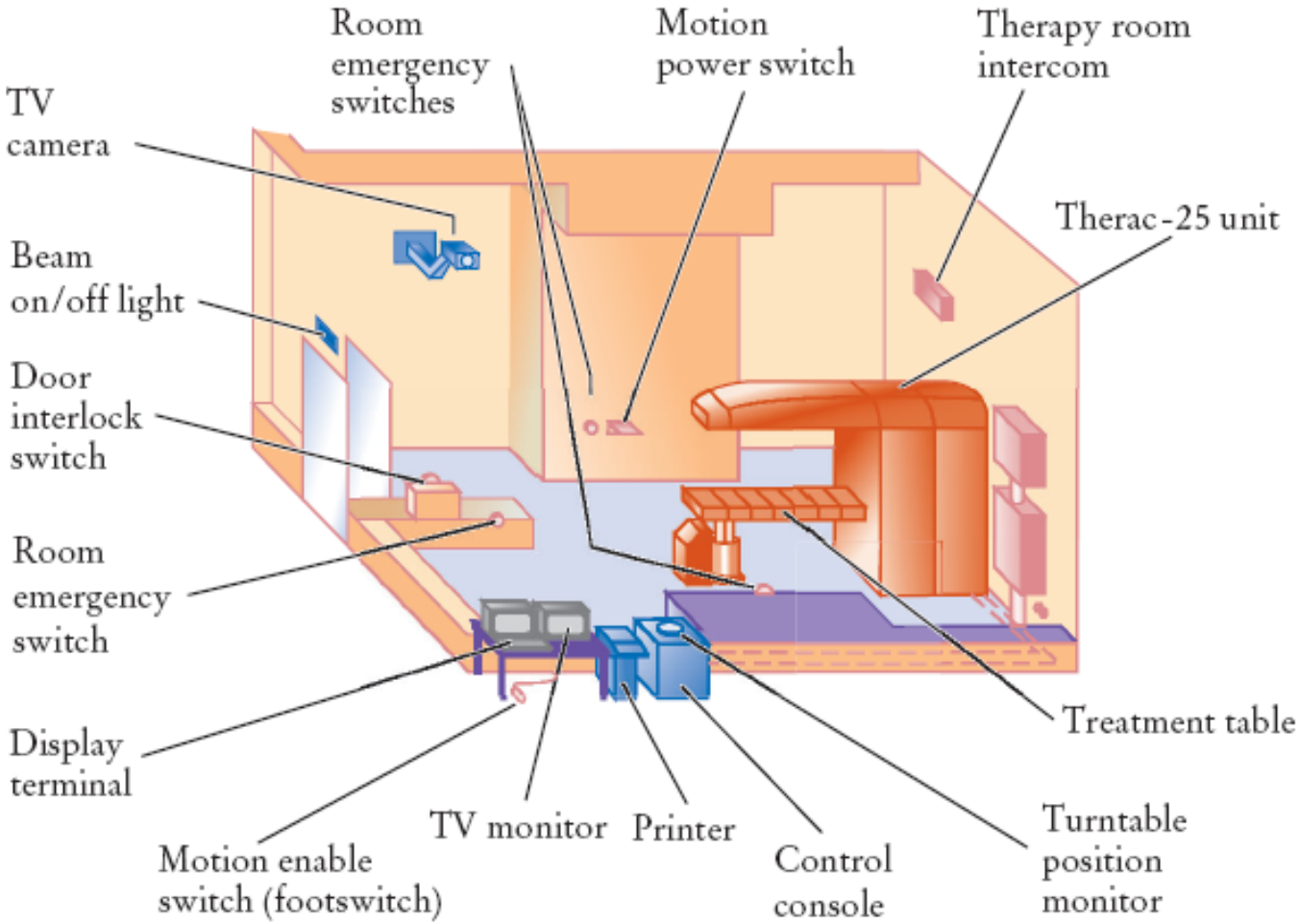
Software: pros and cons

- The pros
  - More functionality (how to develop a GPS or a mobile telephone as an analogue system, without software?)
  - More flexibility (how to change the behavior of an engine, e.g. more power vs. more control and stability, at the push of a button, without software?)
- The cons
  - Software is complex and can introduce defects
  - Software defects can cause system defects and system failures

Therac25



Therac25


Room
emergency
switches
Motion
power switch
Therapy room
intercom
TV
camera
Therac-25 unit
Beam
on/off light
Door
interlock
switch
Room
emergency
switch
Treatment table
Display
terminal
TV monitor
Printer
Turntable
position
monitor
Motion enable
switch (footswitch)
Control
console

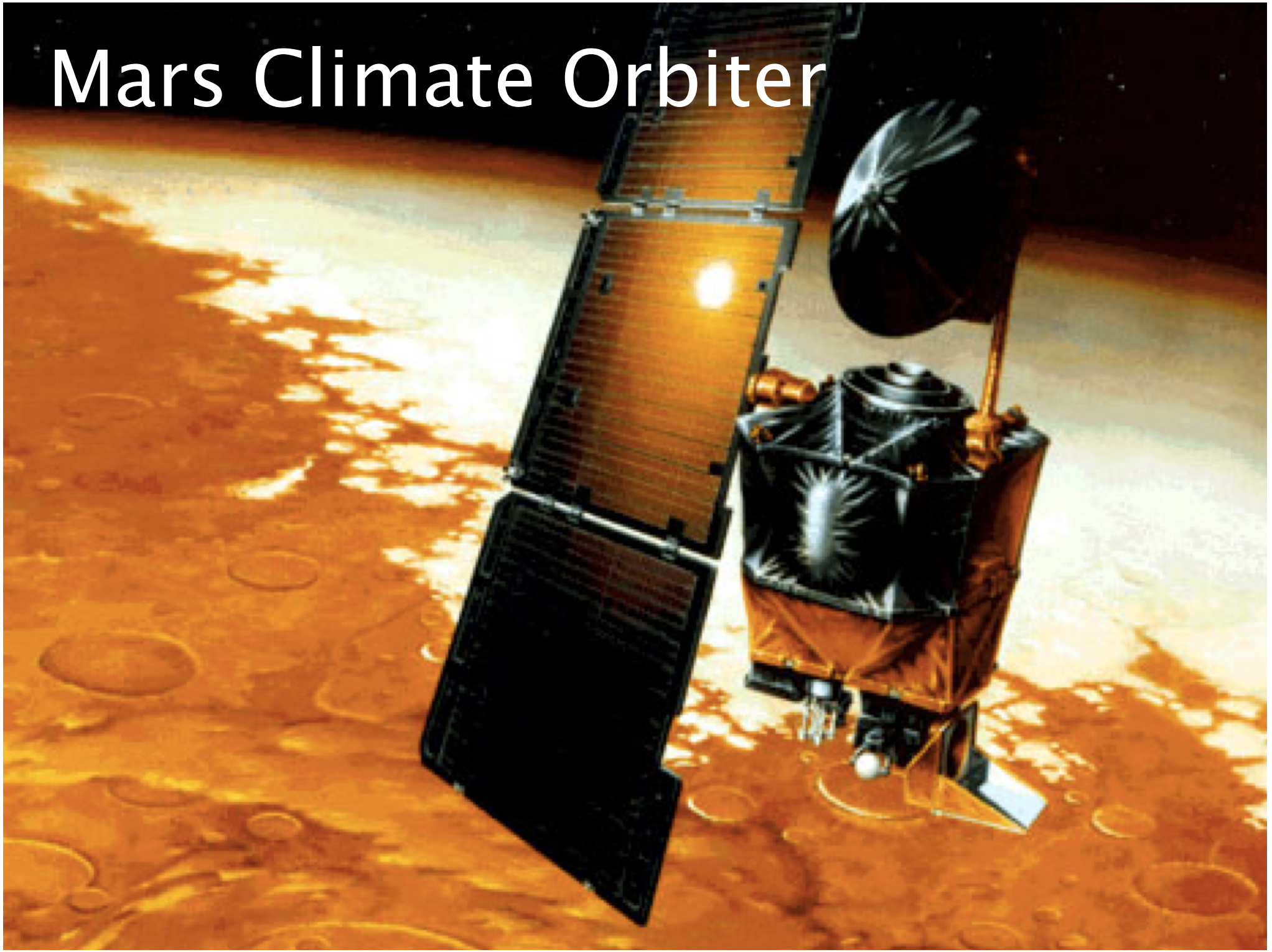
Software failures: safety critical

- Safety critical systems
  - Therac25 – 3 casualties (1985)
    - An X-ray machine to treat cancer
    - A software defect caused excessive treatment time (minutes instead of seconds), burns of many patients and finally death of three of them

Ariane V

Software failures: mission critical

- Ariane V (1996)
  - The European launcher for earth satellites
  - A software defect caused an error in computing the position and speed of the launcher some 30 seconds after launch: the wrong position data caused the controller to send signals to the engines to change the direction of the launcher so swiftly that the structure was subject to high tensions; the tension went over the acceptable thresholds and the safety controller ordered self destruction


Mars Climate Orbiter

Software failures: mission critical

- Mars Climate Orbiter (1999)
  - A probe expected to study atmospheric conditions on Mars
  - A measure of length had to be exchanged between two components developed by two different teams: the two teams used two different unit of measures; the difference was very small and went unnoticed until the probe crashed on Mars
- ATT switching system (1990)
  - The switching system of ATT's long distance telephone system, based on a network of computers
  - A defect on one computer caused shutdown of it, and shutdown of the others directly connected to it; this snowball effect caused large part of the East Coast to be disconnected for many hours

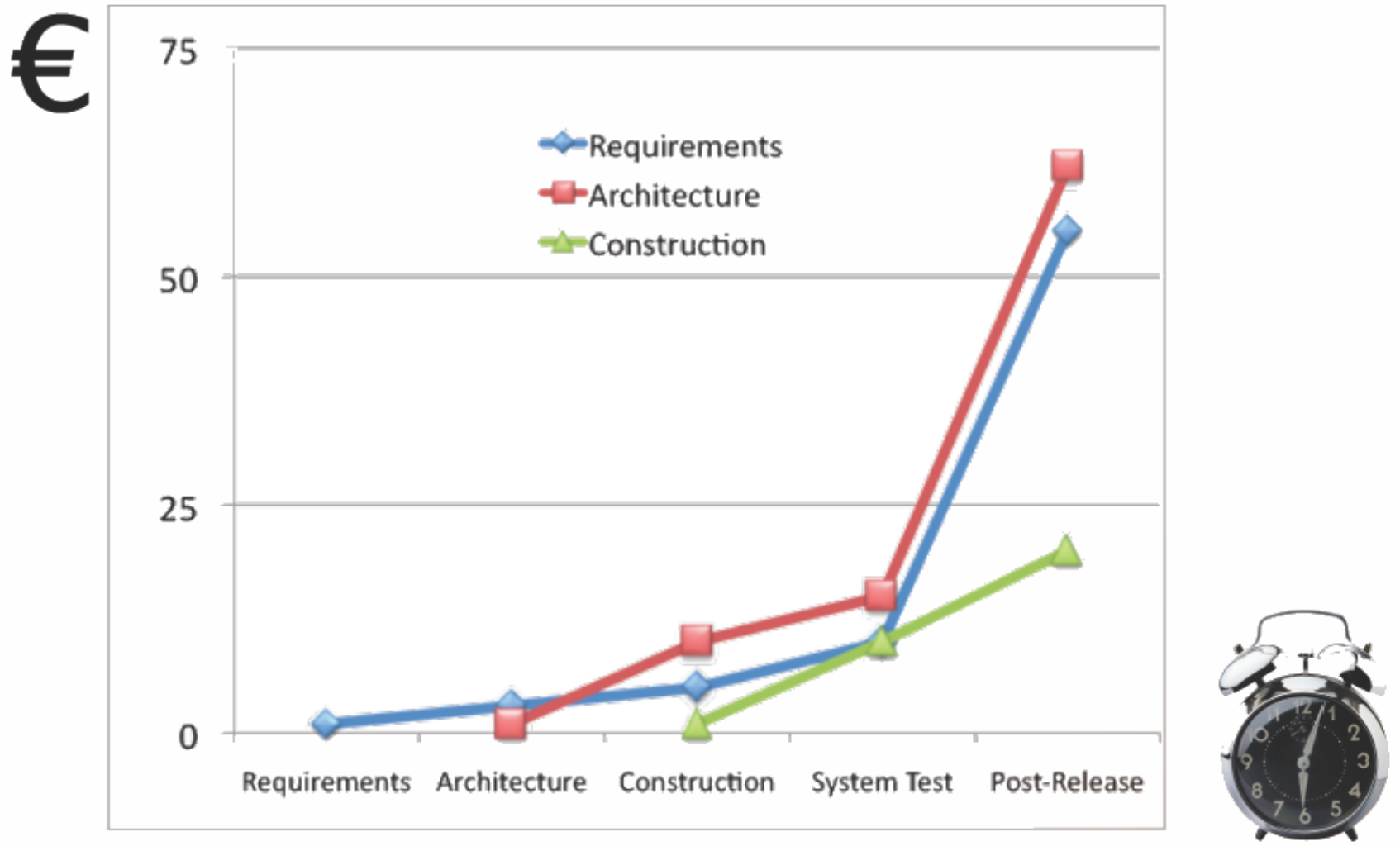
Software costs

- Software costs often dominate computer system costs: the costs of software on a PC are often greater than the hardware cost
- Software costs more to maintain than it does to develop: for systems with a long life, maintenance costs may be several times development costs

Quality

- Most defects injected in requirements and design
- The earlier a defect injected, the more costly to correct it
  - Because defects are usually found in operation

Cost of detection delay



Functional vs. non functional

- Functional characteristics of software
  - "Add two integer numbers"
- Non functional properties
  - User interface usable by not computer expert
  - Precision
    - relative error < 10-9
    - absolute error < 10-8
  - Reliability
    - sum must be correct 99,999999% times
  - Performance, efficiency
    - Sum must be performed < 0,01 millisec
    - Sum must use <10 kbytes ram memory

Functional vs. non functional

- Non functional properties sometimes harder to express
- Harder to design into software
  - They are *emerging* properties
  - Depend on the whole system, i.e. reliability, performance

Myths

- Software is inexpensive
  - Add-on to engineering products, as product often free?
  - Very labor intensive, assuming
    - Productivity = 200 - 1000 LOC per person month
    - Personal cost = : $ 8.000 per person month
    - Total cost: $8 to $40 per LOC
- A medium sized project with 50.000 LOC costs between $400.000 to $1.600.000 in personnel

Typical software problems

- Too expensive (up to a factor of 10)
- Delivered too late (up to a factor of 2)
- Does not live up to user expectations (e.g., reliability)
- High-Profile Example Failures
  - Ariane-5 accident (System process problem)
  - Year-2000 problem (System architecture/documentation problem)

Engineering vs solo programming

- Software large
- Team / teams
- Requirements defined by customer, contract
- Many deliverables
- Many changes, long lifespan
- Costly

- Software small
- One person
- Requirements defined by programmer
- One deliverable
- Few changes, short lifespan
- Free

1968, software crisis

- Term coined at NATO conference in Garmish Partenkirchen
  - Software crisis: the problem
    - The computer industry at large was having a great deal of trouble in producing large and complex software systems
  - Software Engineering (SE): the solution

Software Engineering

- "Multi person construction of multi version software" (Parnas)
- "A discipline that deals with the building of software systems which are so large that they are built by a team or teams of engineers" (Ghezzi, Jazayeri, Mandrioli)

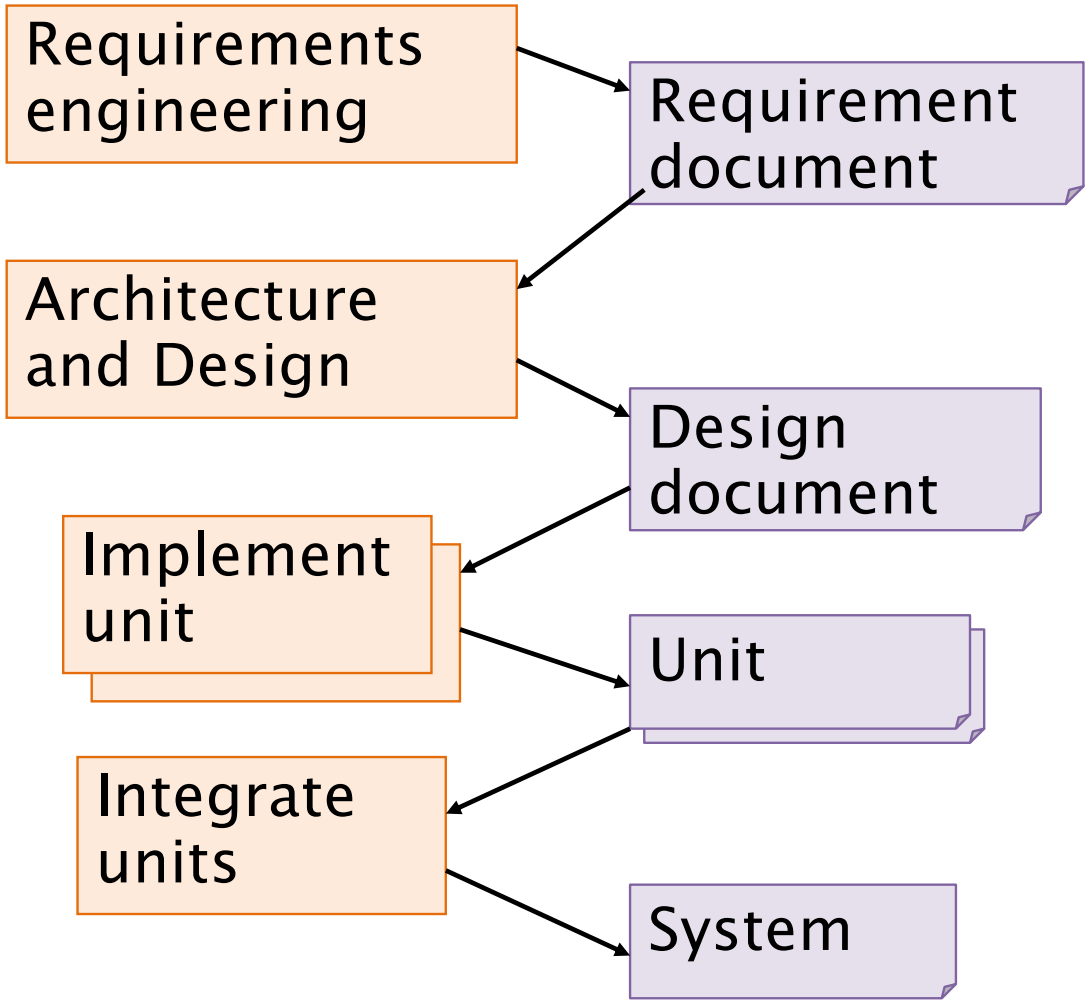
Goal

- Produce software
  - documents, data, code
- … with defined, predictable process properties
  - cost, duration
- … and product properties
  - functionality, reliability, performance, etc.

The production activities

- Requirement engineering
  - What the software should do
- Architecture and design
  - What units and how organized
- Implementation
  - Write source code
  - Integrate units

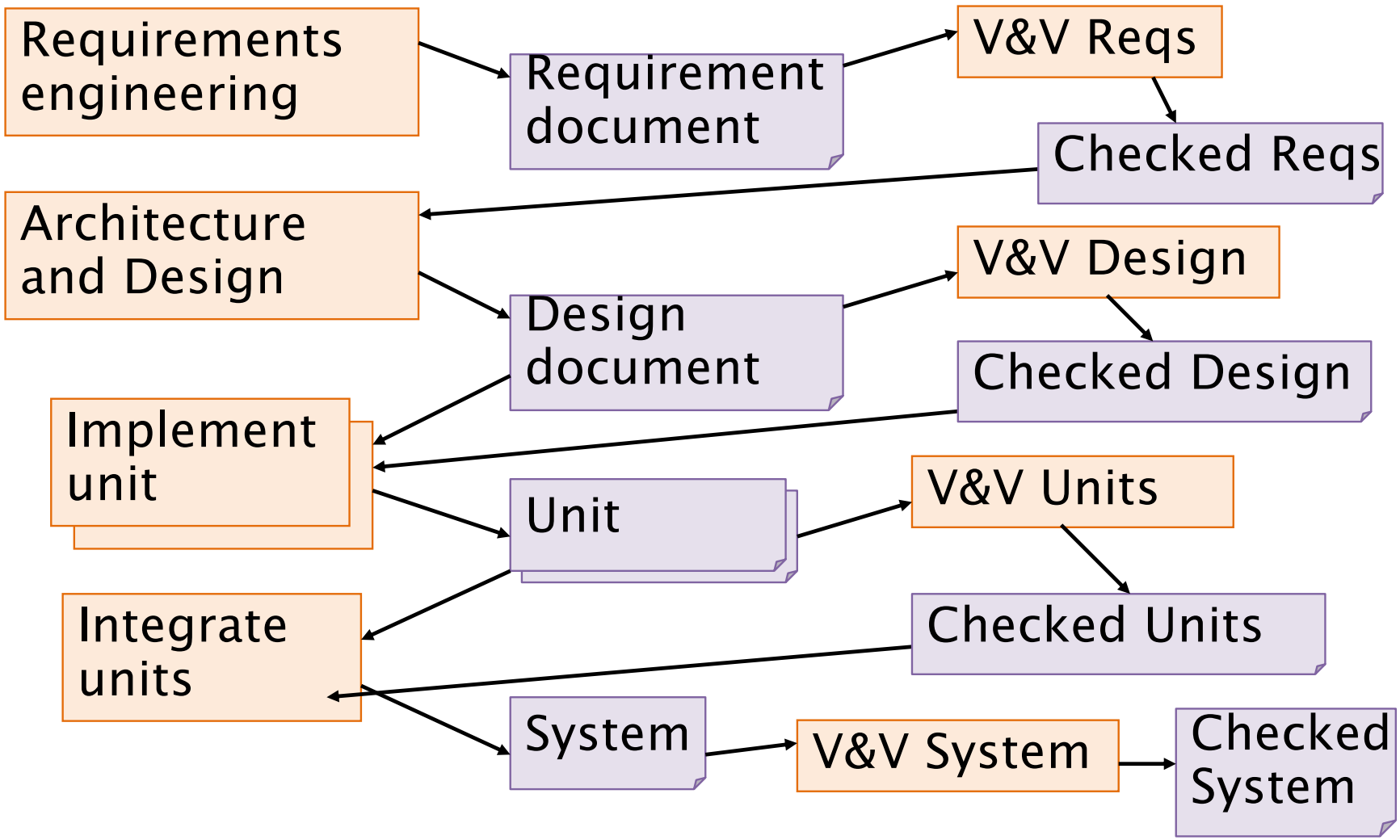
The production activities



V & V

- Verification and validation
  - Control that the requirements are correct
  - Control that the design is correct
  - Control that the code is correct

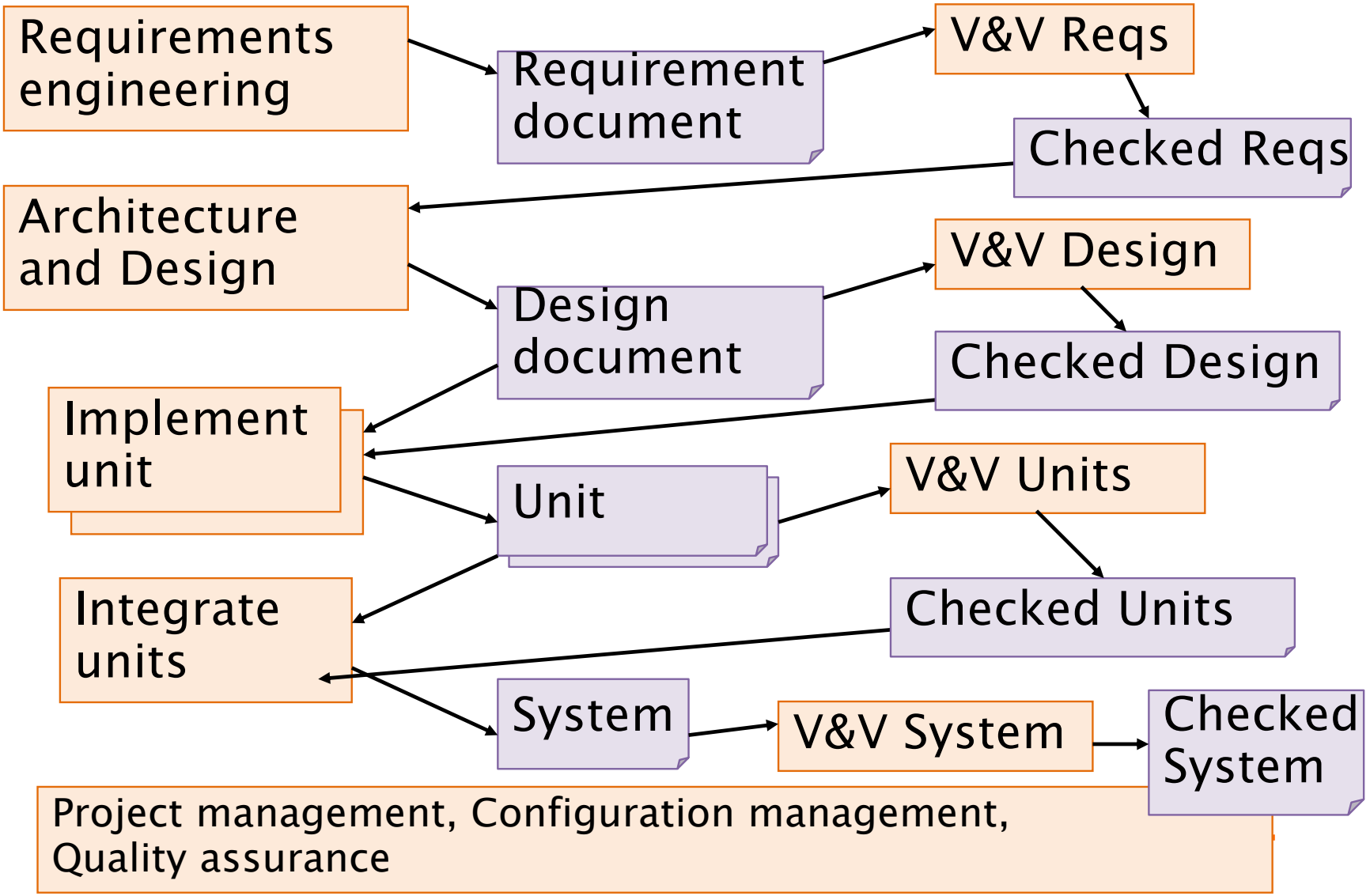
The production activities



Management

- Project management
  - Assign work and monitor progress
  - Estimate and control budget
- Configuration management
  - Identify, store documents and units
  - Keep track of relationships and history
- Quality assurance
  - Define quality goals
  - Define how work will be done
  - Control results

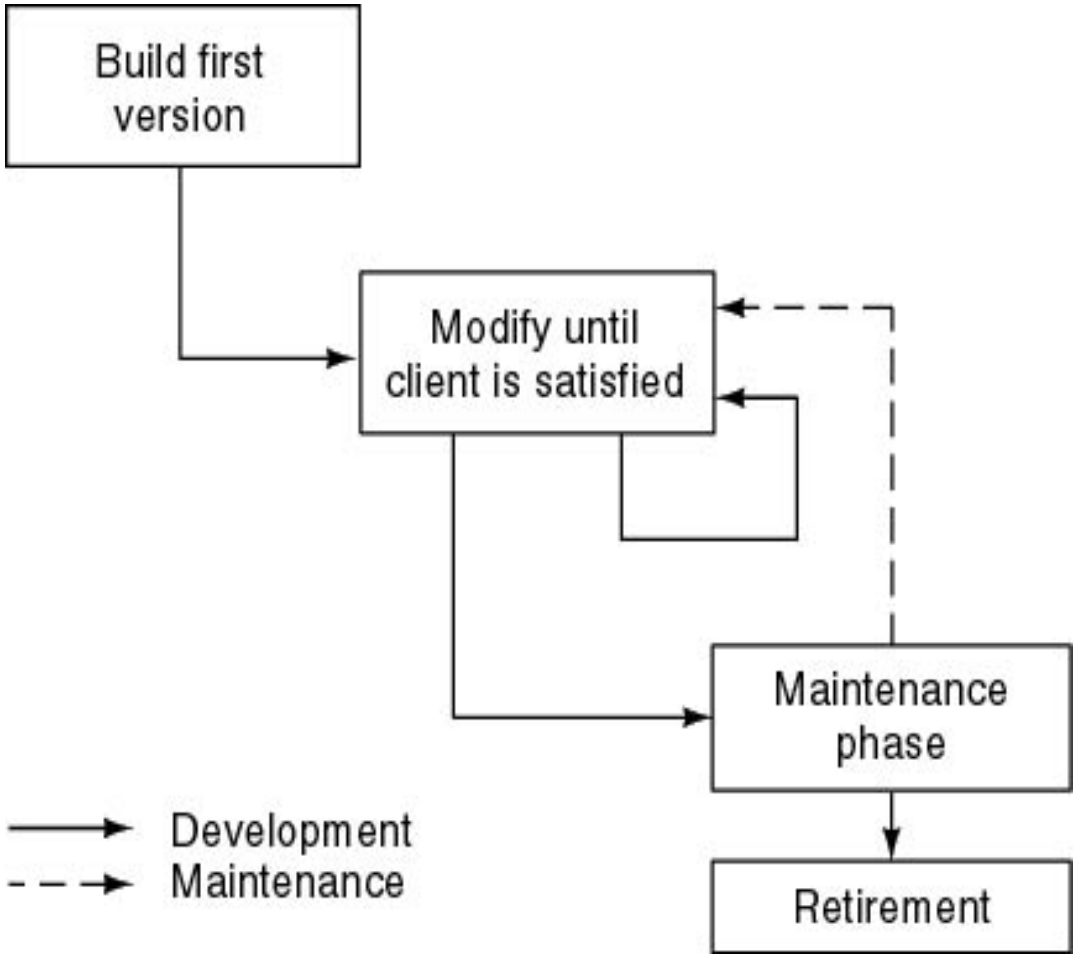
The production activities



Process models

- "Cowboy" programming a.k.a. Build and Fix
  - Just code, all the rest is time lost and real programmers don't do it
- Document-based, semiformal, UML
  - Semiformal language for documents (UML), hand (human) based transformations and controls
- Formal/model based
  - Formal languages for documents, automatic transformations and controls
- Agile
  - Limited use of documents

Build and Fix

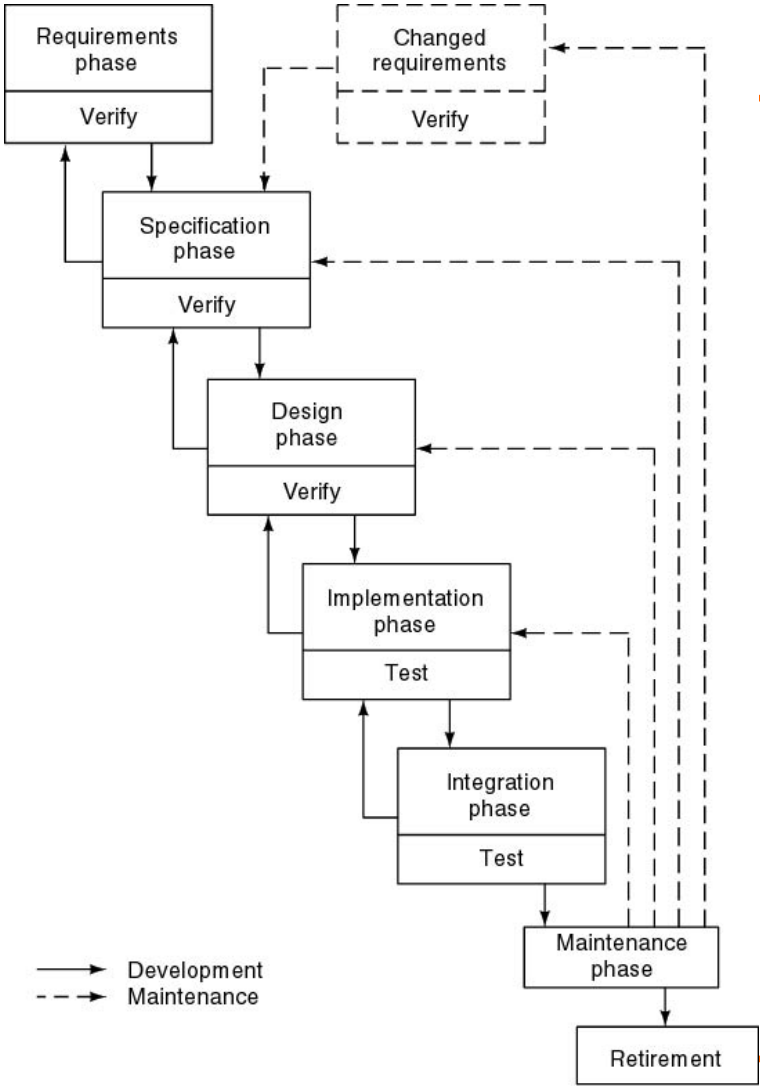


More structured models

- Document-based
  - Waterfall
  - Incremental, Evolutionary, Iterative
  - Prototyping
  - Spiral
  - Unified Process – UP – RUP
- Agile
  - Scrum, Extreme Programming, Crystal
- Formal methods
  - Formal methods
  - Formal UML

Waterfall

- Sequential activities
  - Activity produces document/deliverable
  - Next activity starts when previous is over and freezes the deliverable
  - Change of documents/deliverables is an exceptional case
- Document driven



Agile manifesto – Values

- Individuals and interactions
  - over processes and tools
- Working software
  - over comprehensive documentation
- Customer collaboration
  - over contract negotiation
- Responding to change
  - over following a plan

Agile development principles

- Test as you go
- Deliver product early and often
  - Feedback
- Document as you go, only as required
- Build cross-functional teams

Trends – development

- Agile
- Component based SE
  - Buy + integrate vs. build
  - Open source or commercial
- Offshoring
- Outsourcing