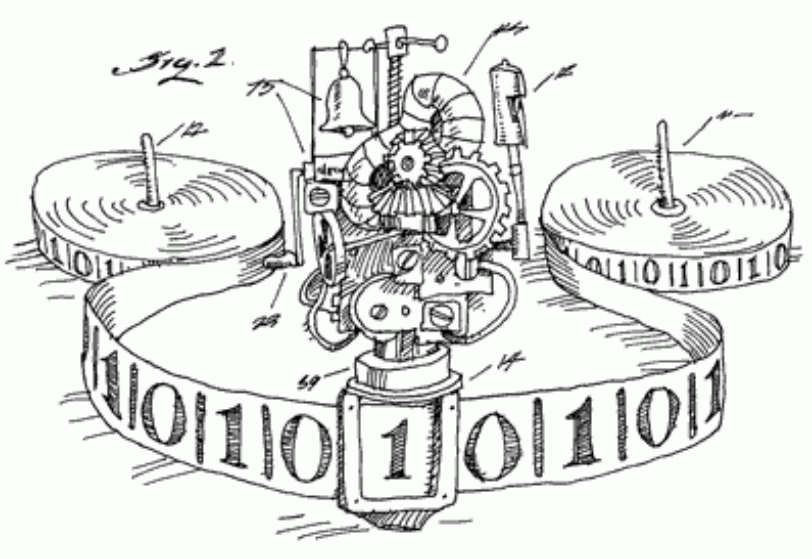
INFO 101 – Introduction to Computing and Security

[2020 - Week 2 / 1]



Prof. Dr. Rui Abreu

University of Porto, Portugal

rui@computer.org

@rmaranhao

What we are going to discuss…

- Introduction to software. What is software? What is software engineering?
- Discussing the interactions between software and hardware
- The Internet and the WWW
- The basic principles of data communication

ASCII vs. UTF-8 vs. UTF-16

- Interesting read about encodings
  - https://medium.com/@apiltamang/unicode-utf-8-and-ascii-encodings-made-easy-5bfbe3a1c45a

> *So I have an announcement to make: if you are a programmer working in 2017 and you don't know the basics of characters, character sets, encodings, and Unicode, and I* catch *you, I'm going to punish you by making you peel onions for 6 months in a submarine. I swear I will.*
>
> *- (Joel Spolsky. Cofounder of stack-overflow. Taken from article here)*

How does a PC represent your name?

Regular ASCII Chart (character codes 0 - 127)

000d	00h		(nul)	016d	10h	▶	(dle)	032d	20h	sp	048d	30h	0	064d	40h	@	080d	50h	P	096d	60h	`	112d	70h	p
001d	01h	☺	(soh)	017d	11h	◀	(dc1)	033d	21h	!	049d	31h	1	065d	41h	A	081d	51h	Q	097d	61h	a	113d	71h	q
002d	02h	☻	(stx)	018d	12h	↕	(dc2)	034d	22h	"	050d	32h	2	066d	42h	B	082d	52h	R	098d	62h	b	114d	72h	r
003d	03h	♥	(etx)	019d	13h	‼	(dc3)	035d	23h	#	051d	33h	3	067d	43h	C	083d	53h	S	099d	63h	c	115d	73h	s
004d	04h	♦	(eot)	020d	14h	¶	(dc4)	036d	24h	$	052d	34h	4	068d	44h	D	084d	54h	T	100d	64h	d	116d	74h	t
005d	05h	♣	(enq)	021d	15h	§	(nak)	037d	25h	%	053d	35h	5	069d	45h	E	085d	55h	U	101d	65h	e	117d	75h	u
006d	06h	♠	(ack)	022d	16h	▬	(syn)	038d	26h	&	054d	36h	6	070d	46h	F	086d	56h	V	102d	66h	f	118d	76h	v
007d	07h	•	(bel)	023d	17h	↨	(etb)	039d	27h	'	055d	37h	7	071d	47h	G	087d	57h	W	103d	67h	g	119d	77h	w
008d	08h	◘	(bs)	024d	18h	↑	(can)	040d	28h	(	056d	38h	8	072d	48h	H	088d	58h	X	104d	68h	h	120d	78h	x
009d	09h		(tab)	025d	19h	↓	(em)	041d	29h	)	057d	39h	9	073d	49h	I	089d	59h	Y	105d	69h	i	121d	79h	y
010d	0Ah		(lf)	026d	1Ah		(eof)	042d	2Ah	*	058d	3Ah	:	074d	4Ah	J	090d	5Ah	Z	106d	6Ah	j	122d	7Ah	z
011d	0Bh	♂	(vt)	027d	1Bh	←	(esc)	043d	2Bh	+	059d	3Bh	;	075d	4Bh	K	091d	5Bh	[	107d	6Bh	k	123d	7Bh	{
012d	0Ch	♀	(np)	028d	1Ch	∟	(fs)	044d	2Ch	,	060d	3Ch	<	076d	4Ch	L	092d	5Ch	\	108d	6Ch	l	124d	7Ch	|
013d	0Dh		(cr)	029d	1Dh	↔	(gs)	045d	2Dh	-	061d	3Dh	=	077d	4Dh	M	093d	5Dh	]	109d	6Dh	m	125d	7Dh	}
014d	0Eh	♫	(so)	030d	1Eh	▲	(rs)	046d	2Eh	.	062d	3Eh	>	078d	4Eh	N	094d	5Eh	^	110d	6Eh	n	126d	7Eh	~
015d	0Fh	☼	(si)	031d	1Fh	▼	(us)	047d	2Fh	/	063d	3Fh	?	079d	4Fh	O	095d	5Fh	_	111d	6Fh	o	127d	7Fh	⌂

Extended ASCII Chart (character codes 128 – 255; Codepage 850)

128d	80h	Ç	144d	90h	É	160d	A0h	á	176d	B0h	▒	192d	C0h	└	208d	D0h	ð	224d	E0h	Ó	240d	F0h	-
129d	81h	ü	145d	91h	æ	161d	A1h	í	177d	B1h	▒	193d	C1h	┴	209d	D1h	Ð	225d	E1h	ß	241d	F1h	±
130d	82h	é	146d	92h	Æ	162d	A2h	ó	178d	B2h	▓	194d	C2h	┬	210d	D2h	Ê	226d	E2h	Ô	242d	F2h	‗
131d	83h	â	147d	93h	ô	163d	A3h	ú	179d	B3h	|	195d	C3h	├	211d	D3h	Ë	227d	E3h	Ò	243d	F3h	¾
132d	84h	ã	148d	94h	ö	164d	A4h	ñ	180d	B4h	┤	196d	C4h	─	212d	D4h	È	228d	E4h	õ	244d	F4h	¶
133d	85h	à	149d	95h	ò	165d	A5h	Ñ	181d	B5h	Á	197d	C5h	┼	213d	D5h	ı	229d	E5h	Õ	245d	F5h	§
134d	86h	å	150d	96h	û	166d	A6h	ª	182d	B6h	Â	198d	C6h	ã	214d	D6h	Í	230d	E6h	µ	246d	F6h	÷
135d	87h	ç	151d	97h	ù	167d	A7h	º	183d	B7h	À	199d	C7h	Ã	215d	D7h	Î	231d	E7h	þ	247d	F7h	¸
136d	88h	ê	152d	98h	ÿ	168d	A8h	¿	184d	B8h	©	200d	C8h	╚	216d	D8h	Ï	232d	E8h	Þ	248d	F8h	°
137d	89h	ë	153d	99h	Ö	169d	A9h	®	185d	B9h	╣	201d	C9h	╔	217d	D9h	┘	233d	E9h	Ú	249d	F9h	¨
138d	8Ah	è	154d	9Ah	Ü	170d	AAh	¬	186d	BAh	║	202d	CAh	╩	218d	DAh	┌	234d	EAh	Û	250d	FAh	·
139d	8Bh	ï	155d	9Bh	ø	171d	ABh	½	187d	BBh	╗	203d	CBh	╦	219d	DBh	▮	235d	EBh	Ù	251d	FBh	¹
140d	8Ch	î	156d	9Ch	£	172d	ACh	¼	188d	BCh	╝	204d	CCh	╠	220d	DCh	▄	236d	ECh	ý	252d	FCh	³
141d	8Dh	ì	157d	9Dh	Ø	173d	ADh	¡	189d	BDh	¢	205d	CDh	═	221d	DDh	|	237d	EDh	Ý	253d	FDh	²
142d	8Eh	Ä	158d	9Eh	×	174d	AEh	«	190d	BEh	¥	206d	CEh	╬	222d	DEh	Ì	238d	EEh	¯	254d	FEh	■
143d	8Fh	Å	159d	9Fh	ƒ	175d	AFh	»	191d	BFh	┐	207d	CFh	¤	223d	DFh	▀	239d	EFh	´	255d	FFh	

Hexadecimal to Binary

0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

Groups of ASCII-Code in Binary

Bit 6	Bit 5	Group
0	0	Control Characters
0	1	Digits and Punctuation
1	0	Upper Case and Special
1	1	Lower Case and Special

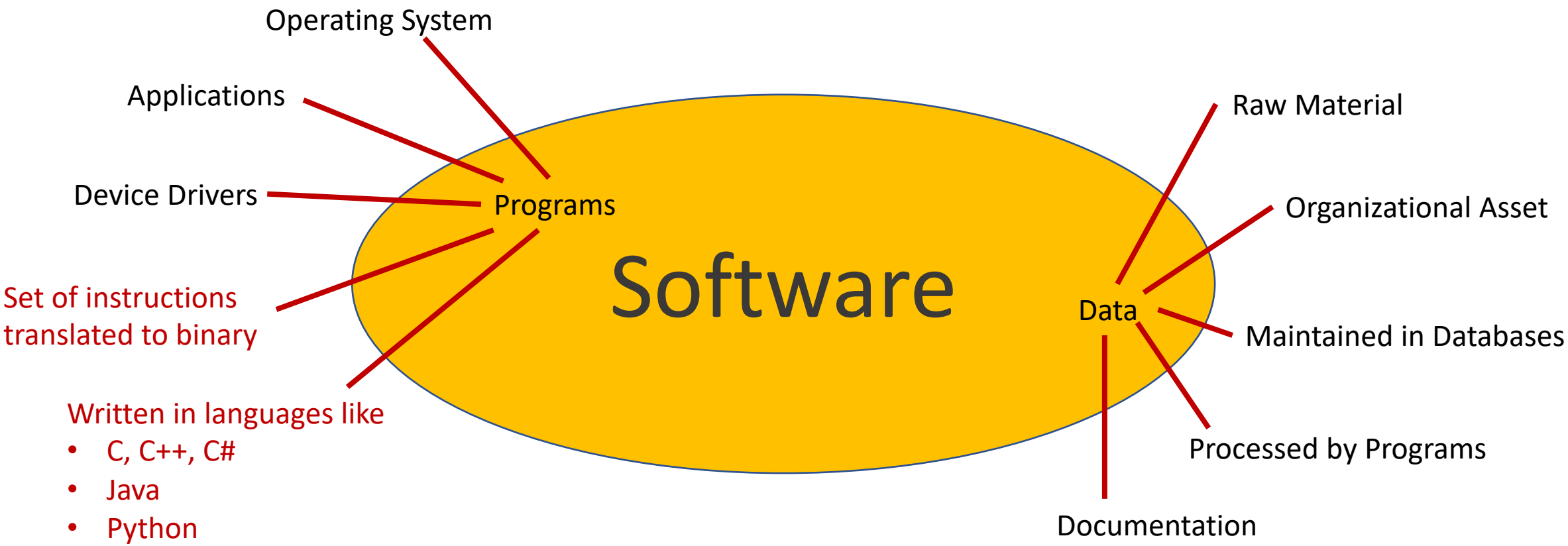
Michael Goerz, Sept 04 2000

Software

What is Software?

*Software – Software is **data and computer programs**. It is not physical and cannot be directly seen. Computer **programs execute within** the **hardware** and enable a computer to perform specific tasks. Programs include **system software** such as an **operating system**, which enables other software to run properly and **application software** such as a word processor or database manager, which **enables a user to perform a task**.*

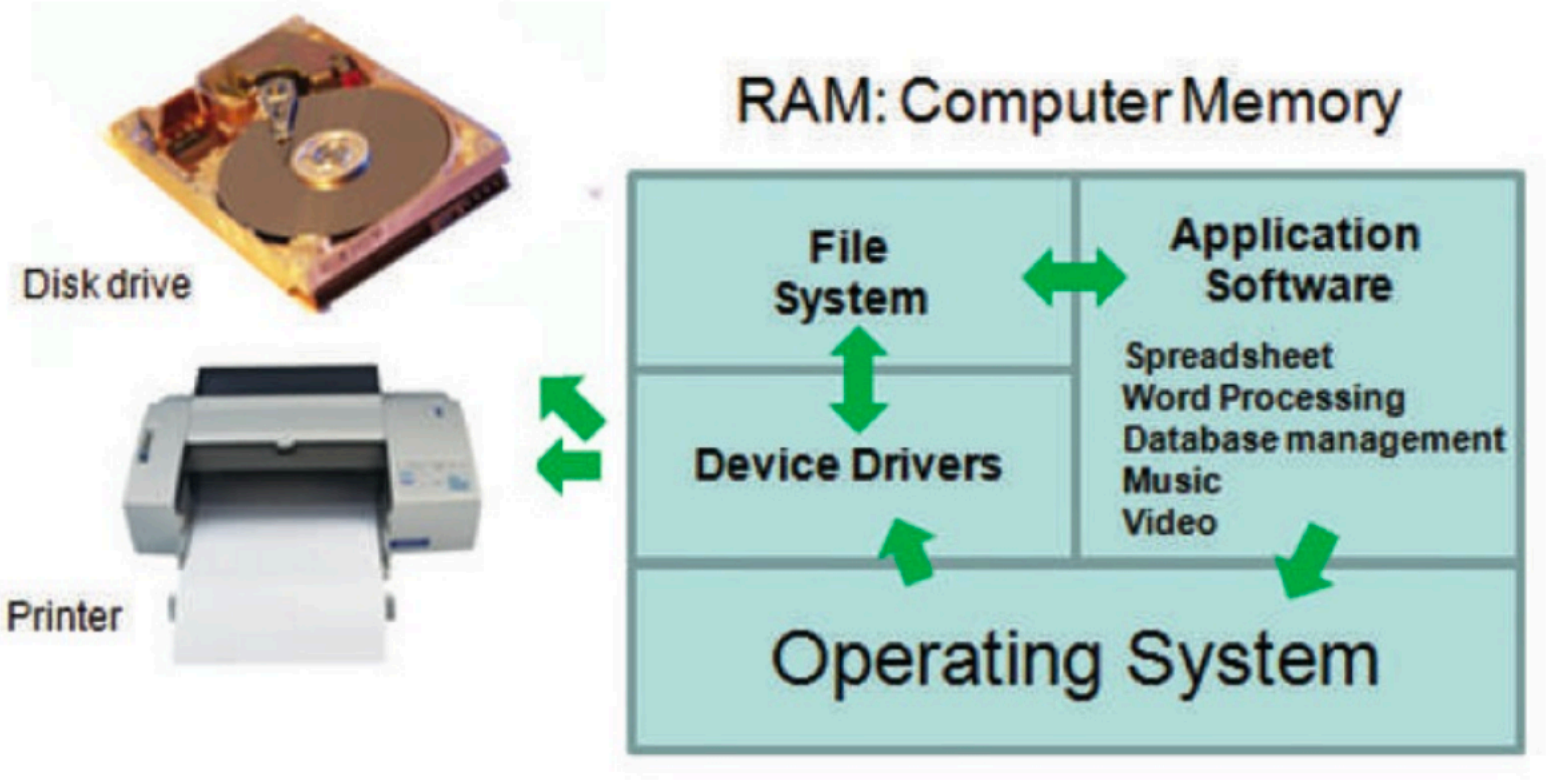
What is Software?



Software & Hardware

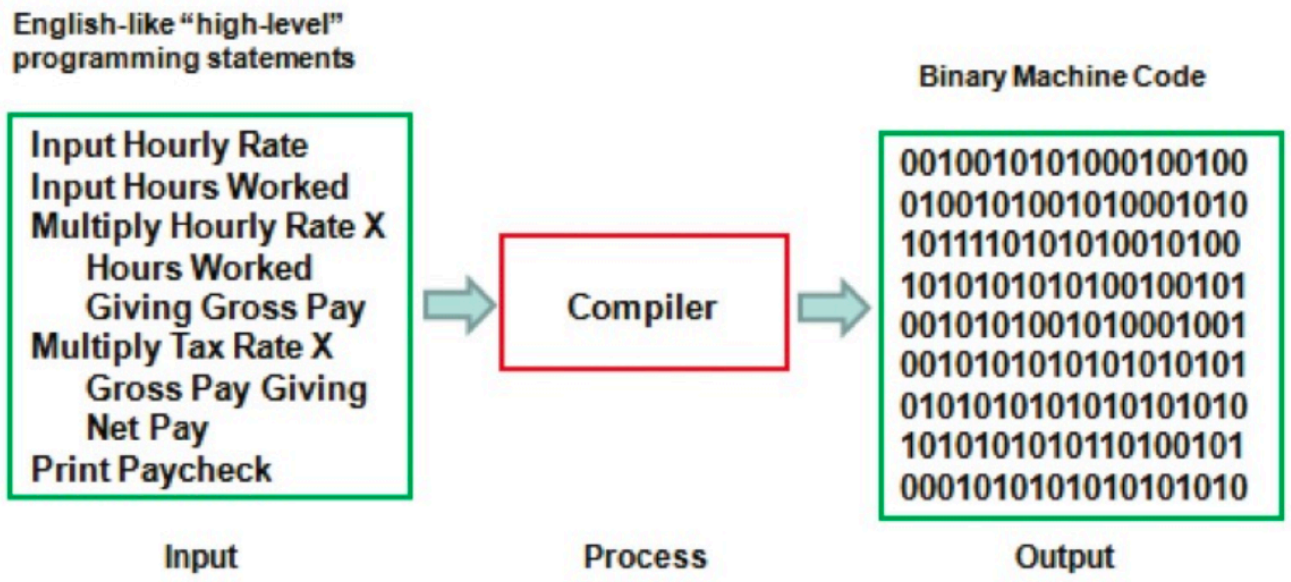
- Software directs the actions of hardware
- Software is loaded into memory (RAM)
- Software is executed in the central processing unit (CPU)
- Different types of software
  - Operating Systems (OS) determines:
    - How to load and executed application programs?
    - How to access files and how to establish communication with the file system?
    - How to access hardware? (Device Drivers)
  - Application Software like
    - Word, Excel, Draw, ... but also Database servers, etc.

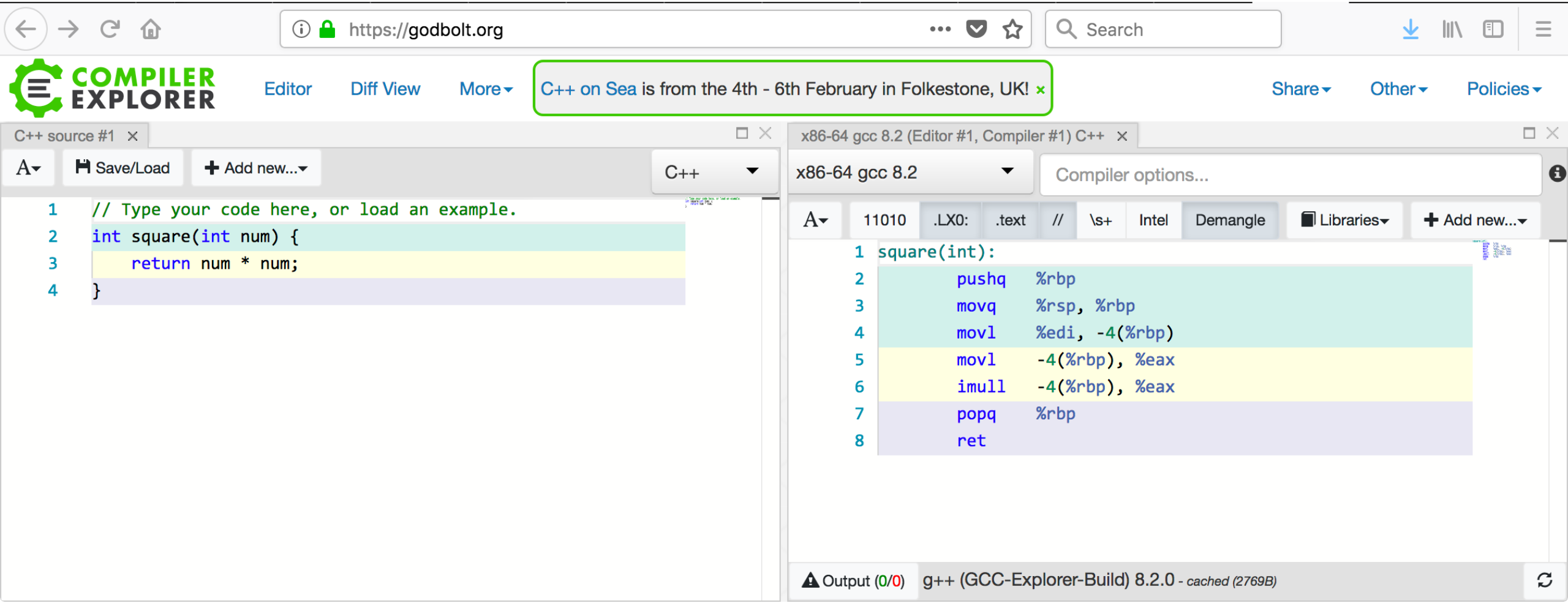
Software & Hardware



Software & Computer Programming

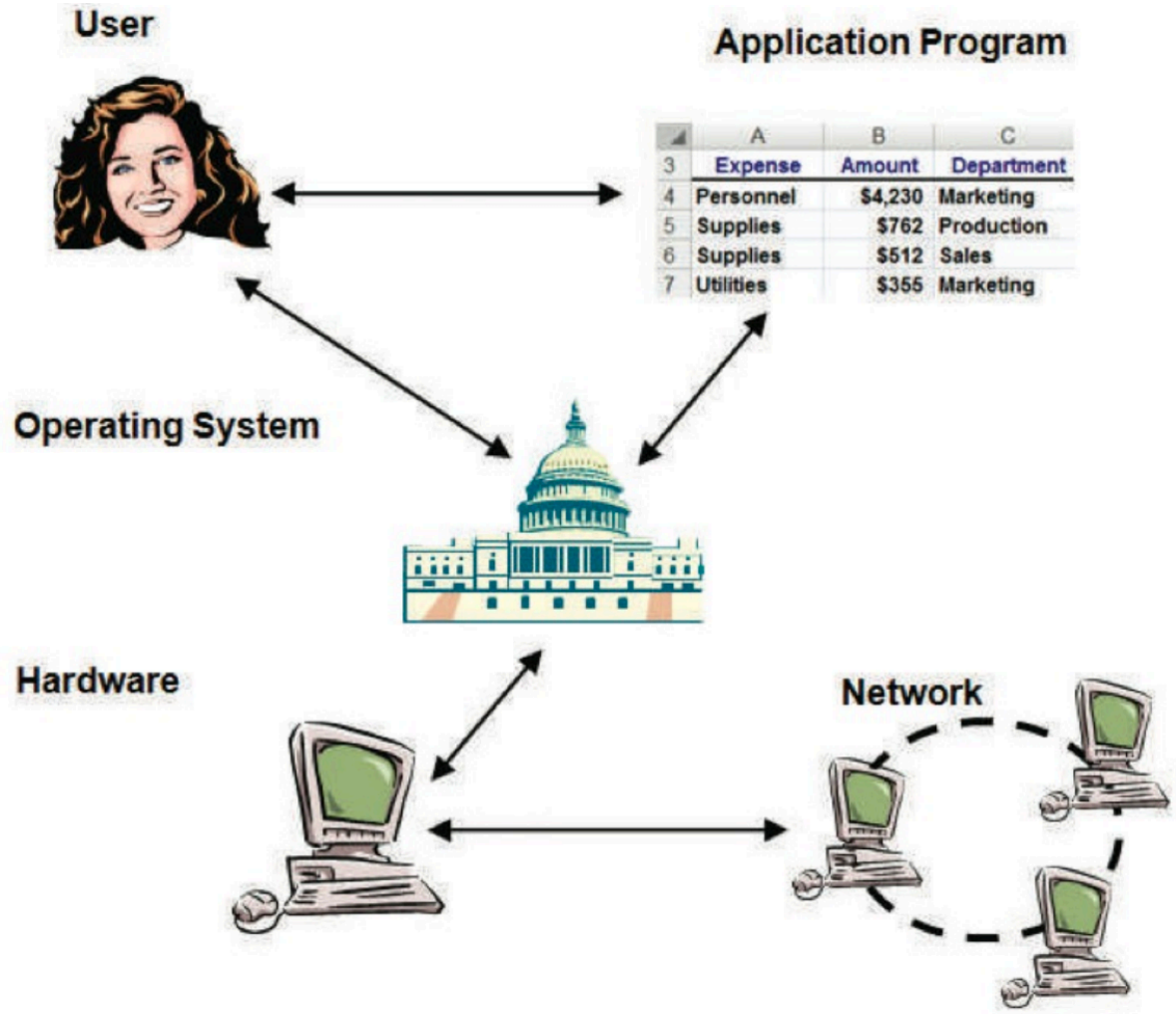
- Software often written in **high-level** programming languages
  - Easier to read, understand, develop and maintain
  - Compilers are used to translate such programs into machine code (sequences of instructions/statements that can be interpreted by the CPU coded in binary format) also called object code




https://godbolt.org
Search
COMPILER EXPLORER
Editor
Diff View
More
C++ on Sea is from the 4th - 6th February in Folkestone, UK!
Share
Other
Policies
C++ source #1
Save/Load
Add new...
C++
1 // Type your code here, or load an example.
2 int square(int num) {
3 return num * num;
4 }
x86-64 gcc 8.2 (Editor #1, Compiler #1) C++
x86-64 gcc 8.2
Compiler options...
11010
.LX0:
.text
//
\s+
Intel
Demangle
Libraries
Add new...
1 square(int):
2 pushq %rbp
3 movq %rsp, %rbp
4 movl %edi, -4(%rbp)
5 movl -4(%rbp), %eax
6 imull -4(%rbp), %eax
7 popq %rbp
8 ret
Output (0/0)
g++ (GCC-Explorer-Build) 8.2.0 - cached (2769B)

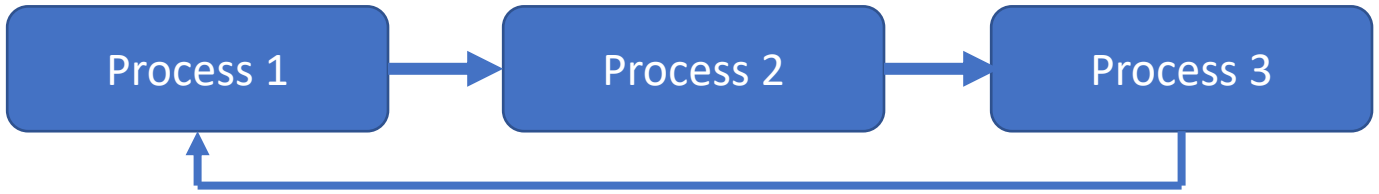
The Operation System

- Controls all computer resources:
  - Memory
  - CPU
  - Access to HW devices
- **Remark**: BIOS (Basic Input/Output System) is a firmware running after starting the computer for initialization and starting the OS



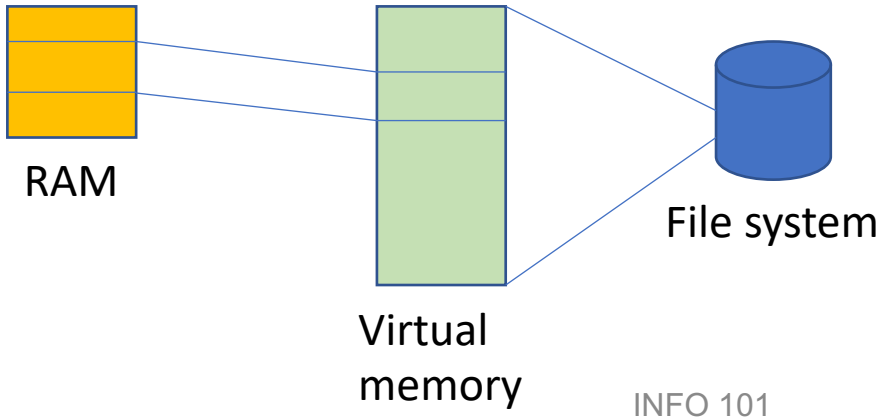
Tasks of an OS

- **Process management**
  - Ever programs runs as a process (or a collection of processes) and needs resources like CPU time and memory
  - The OS handles which process to execute at which time
  - Multi tasking: Processes are executed in parallel
    - Each process has an assigned time
    - OS handles context switching (i.e., the switching between processes)
    - If not enough CPU resources available, the system will trashing. The OS should handle this well.



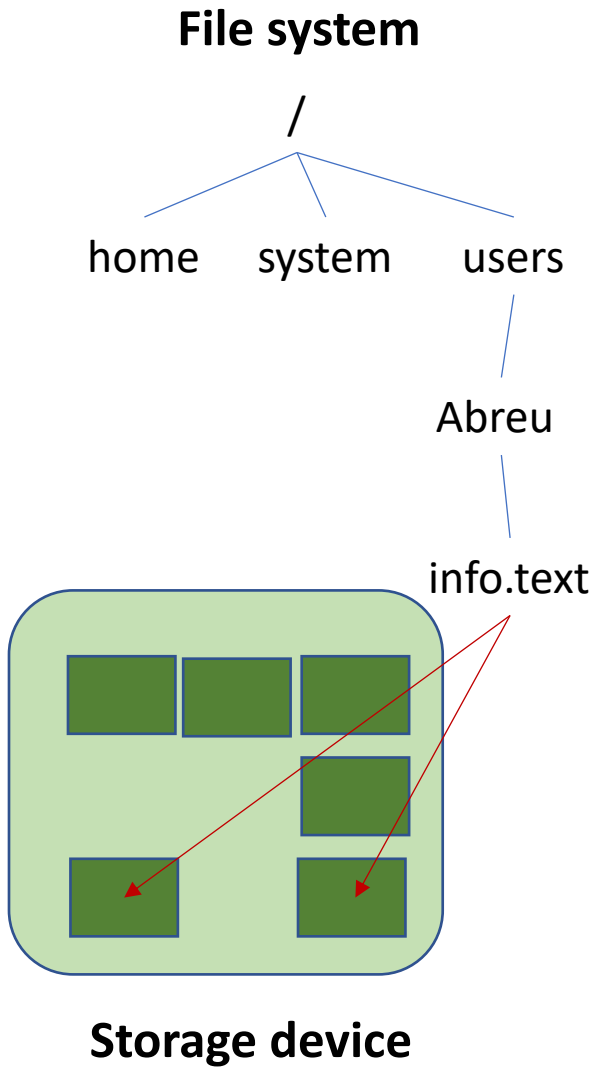
Tasks of an OS

- **Memory management**
  - Assign/allocate memory (RAM) to each program / process
  - Memory is allocated and deallocated whenever needed
  - Handles **virtual memory**:
    - Instead of only using the RAM, the OS can move parts of the content to a file system. Whenever needed, the content is moved from or to the file system
    - Allows to handle memory that is larger than the physical memory



Task of an OS

- **File System**
  - Handles files and directories stored on hard disks etc.
  - Loads file content into memory or stores memory content in a file
  - Keeping track of the physical position of files at storage devices
  - Handles the way files are stored physically including defragmentation
  - Content on storage devices usually is stored in **blocks**
  - **Mapping**: Navigation through the file system, which is **logical** and **not physical**.
  - **File associations**: extensions of files to indicate that they are of the same type (e.g., DOCX, TXT,…)



Task of an OS

- **Networking**
  - Providing Internet access via TCP/IP
  - Work as clients or server
- **Security**
  - **Authentication**: user login to prevent unwanted access
  - **Intrusion detection and prevention**: Prevent from unauthorized intrusion (e.g. using viruses, Trojan horses, or Spyware)
- **Graphical User Interfaces**
- **Device drivers**

Software Ownership

- **Copyright** – **Protect software** to be not used if not granted. **Ownership** of software usually **remains with the author** and **users license software** for their use on machines they own or operate.
- **License** – A license grants permission to a user to use the computer software. Licenses may be permanent or may terminate or may restrict the number of users.
- **Site License** – used to license software to organizations (business, educational, governmental). Grants permission to make a large or unrestricted number of installations throughout the organization.

Software Ownership

- **End User License Agreement** (EULA) – Usually presented during the installation process. Typically an EULA **grants the right** to make a certain number of **copies of the licensed software** among other restrictions and rights.
- **Beta License** – The license of a beta version of a software distributed for the **purpose of evaluation and real-world testing**. The users of a beta version are said **beta testers**. They receive the software **for free** or for a reduced price. **Normally permission is granted for a period of time** that ends when the software reaches the commercial production stage.

Software Ownership

- **Shareware –** Is a method for marketing commercial software in which a **trial version** is distributed in advance and **without payment**. Typically the **trial period is limited** by time or usage or both. At the end of the period the user must decide to **purchase a user license** or allow the software to deactivate.
- **Freeware –** Is specifically licensed to **grant** the recipient **freedom to modify and redistribute** the software. Normally that would be prohibited by copyright law, so the copyright holder of free software must give recipients explicit permission by license to modify and redistribute. A variation in freeware is open source software.

Software Ownership

- **Open Source –** Software whose program source code is available under a copy-right license that permits users to review, alter, and enhance the software, and redistribute it as modified.
- **Software Piracy –** Copyright infringement of software refers to a collection of actions performed without the permission of the copyright holder. Actions include:
  - Creating a copy and selling or giving it to someone.
  - Creating a copy to serve as a backup.
  - Renting the original software to others.
  - Posting copies on the Internet for users to download.

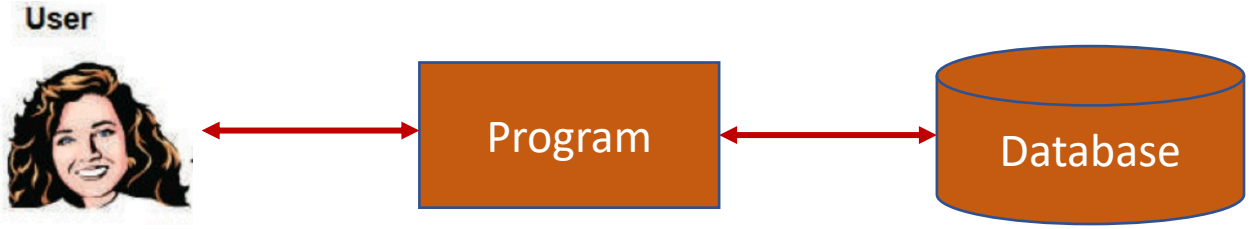
Sometimes, this is not an infringement depending on the national laws

Software Architectures

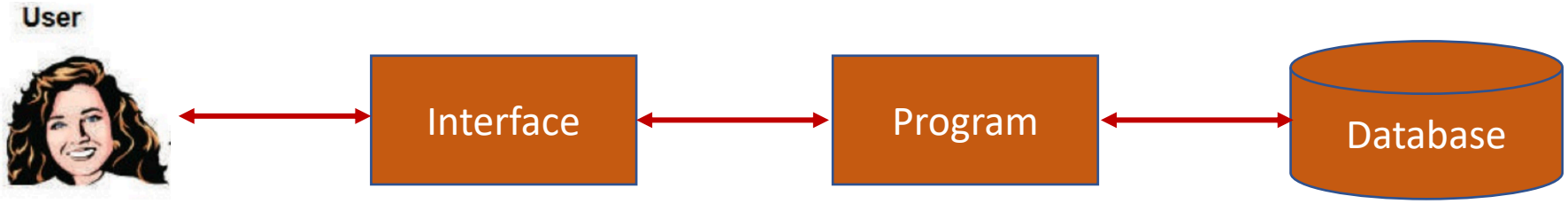
*"Software architecture is the **structure and arrangements of the components of an application**. It also includes the interface requirements of a program, and relationship of the program to database managers and to communication utilities."*

Different software architectures

- Two-Tier architectures
  - Two layers, e.g., application program to database management relationship

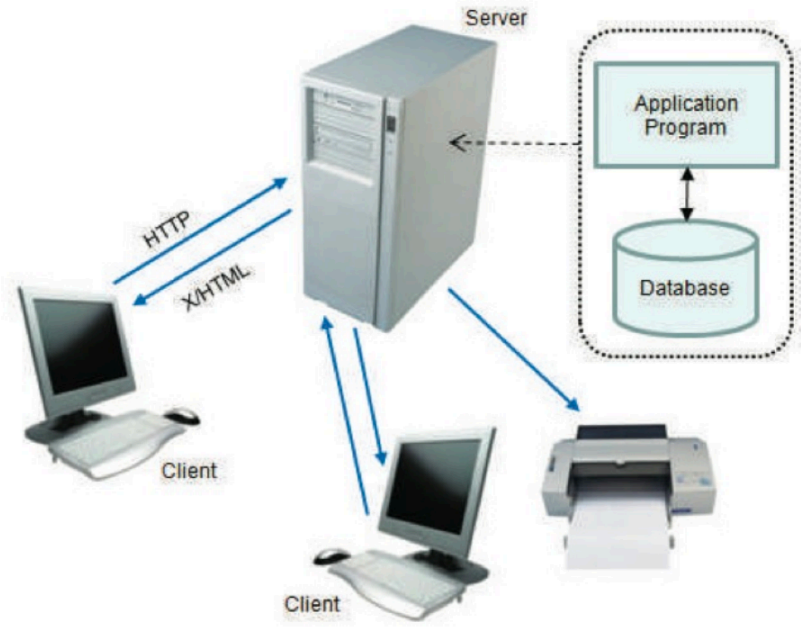


- Three-Tier architectures
  - Three layers, e.g., application program user interface relationship



Different software architectures

- **Client-Server** – separates the client (the consumer of data and other resources) from the server (which distributes those resources). **Examples**: E-Mail services, shared hardware, e.g., printer services,



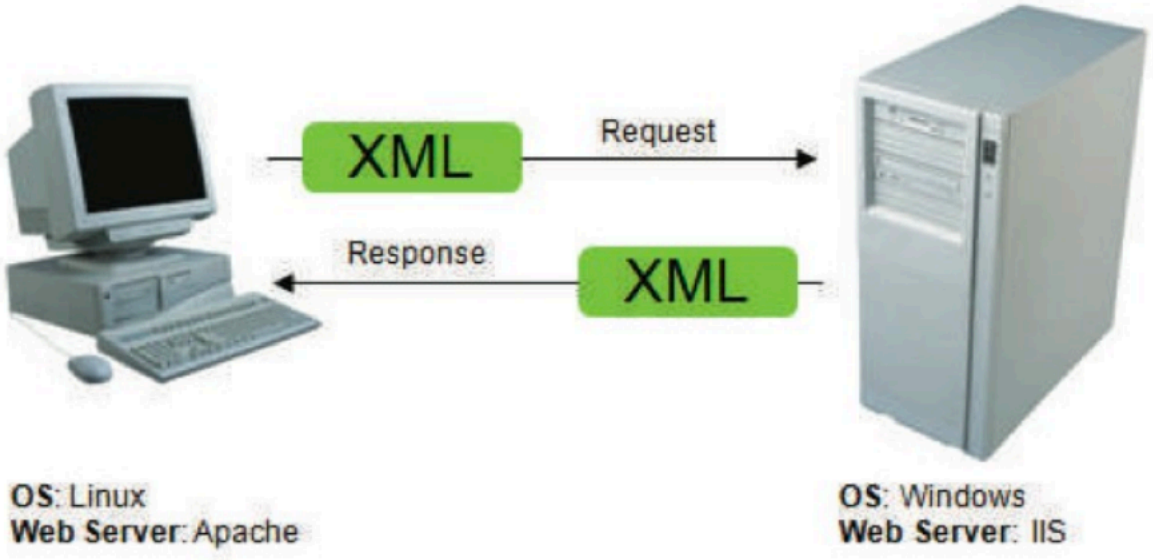
Different software architectures

- **.Net** (Microsoft) – **run-time environment** that manages a program's requirements while standardizing the definitions of data storage structures in memory and providing a **standard interface between software components**.

- **Web Services** – support interoperable between machines over a network. Interoperability is important when machines run different operating systems and different Web servers. Program functionality and data may be distributed over a network of incompatible computers.

Web Services

- One important implementation makes use of XML to communicate content following the **SOAP** (Simple Object Access Protocol) architecture.

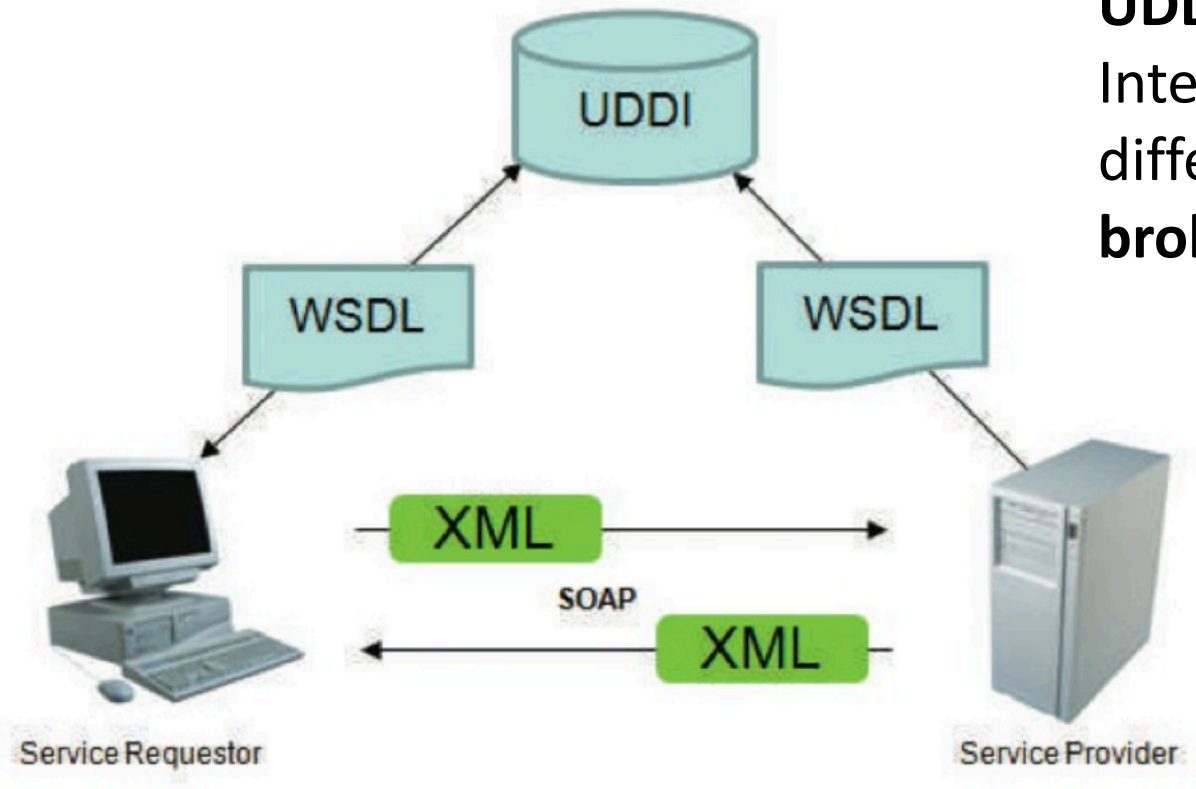


SOAP is a protocol that defines the format of messages exchanged in XML envelopes. HTTP is a foundation for SOAP. The most common format is the Remote Procedure Call (RPC) in which one network node (the client) sends a request message to another node (the server), and the server sends a response message to the client.

Web Services

- Since Web services are distributed among servers it is useful for software developers to have access to a directory of the technical specifications of available services. **WSDL (Web Services Description Language)** is one such directory. It is an XML-based service providing descriptions on how to communicate using Web services. The directory provides information about what **services are available** on **which servers**. It also identifies the **data requirements** for using each service. Programs on the client computer format requests using SOAP to actually call the functions listed in the WSDL.

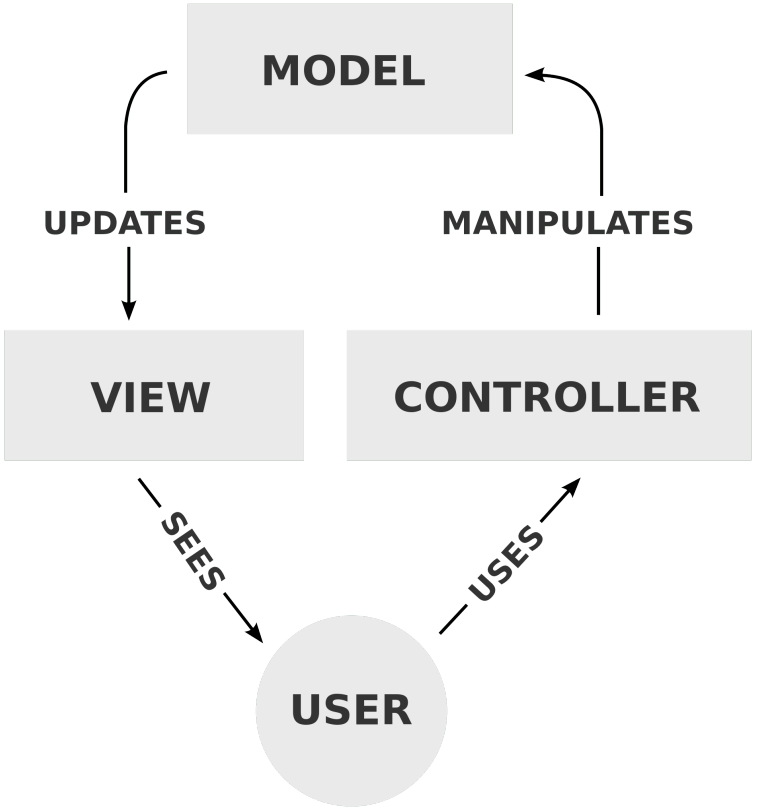
Web Services



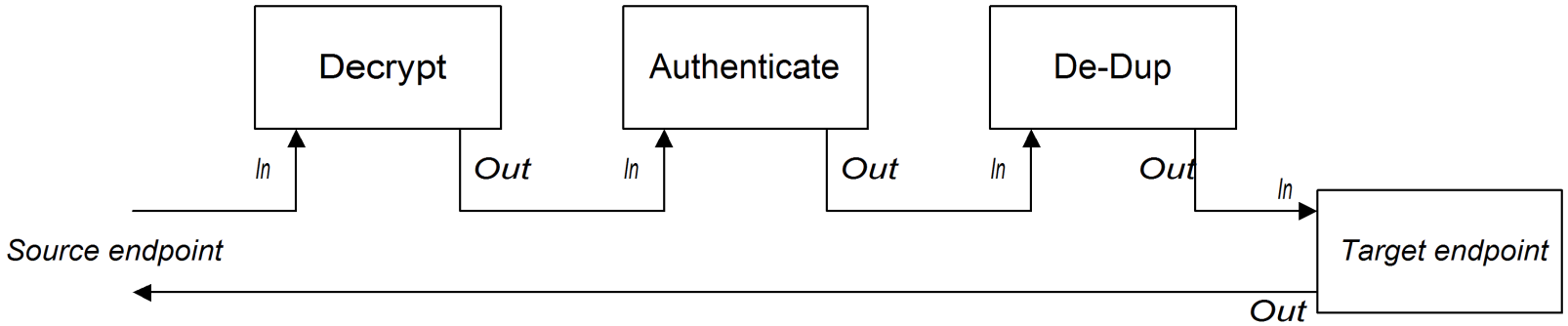
UDDI (Universal Description, Discovery and Integration) lists available Web services from different vendors. UDDI can be seen as a **broker** between businesses.

Other software architectures

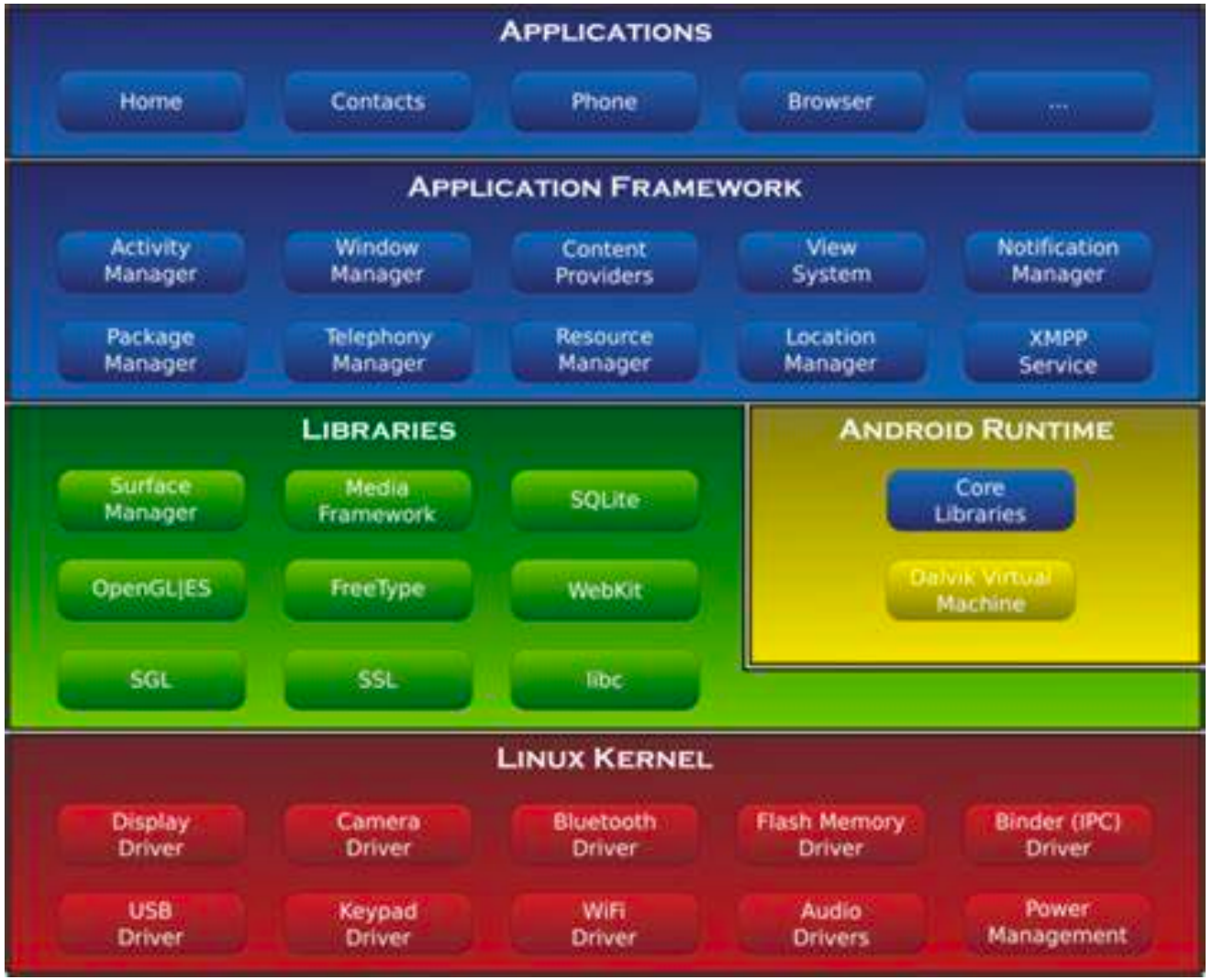
- **Model-View-Controller** for graphical user interfaces (GUIs)



- **Pipes-and-Filters** for connecting different functions to form a more general program



Android Software Architecture



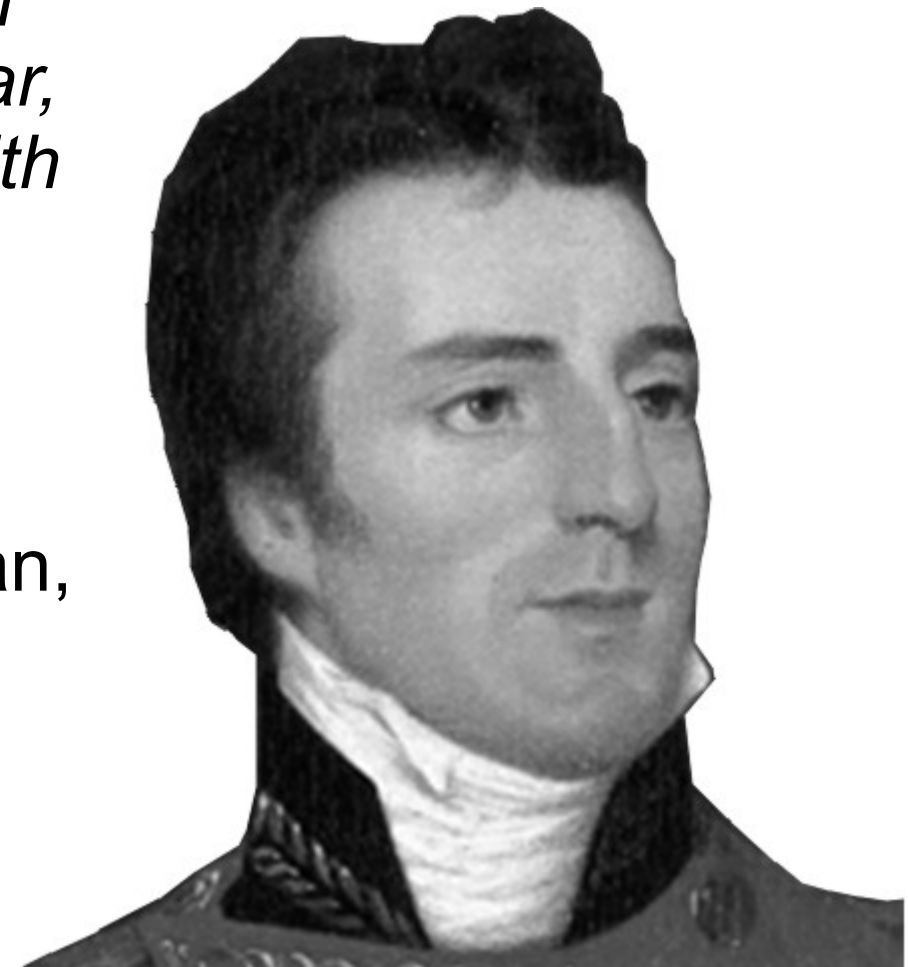
Software Engineering

General principles

A Quote on Engineering

"To define it rudely but not ineptly, ENGINEERING is the art of doing that well with one dollar, which any bungler can do with two after a fashion."

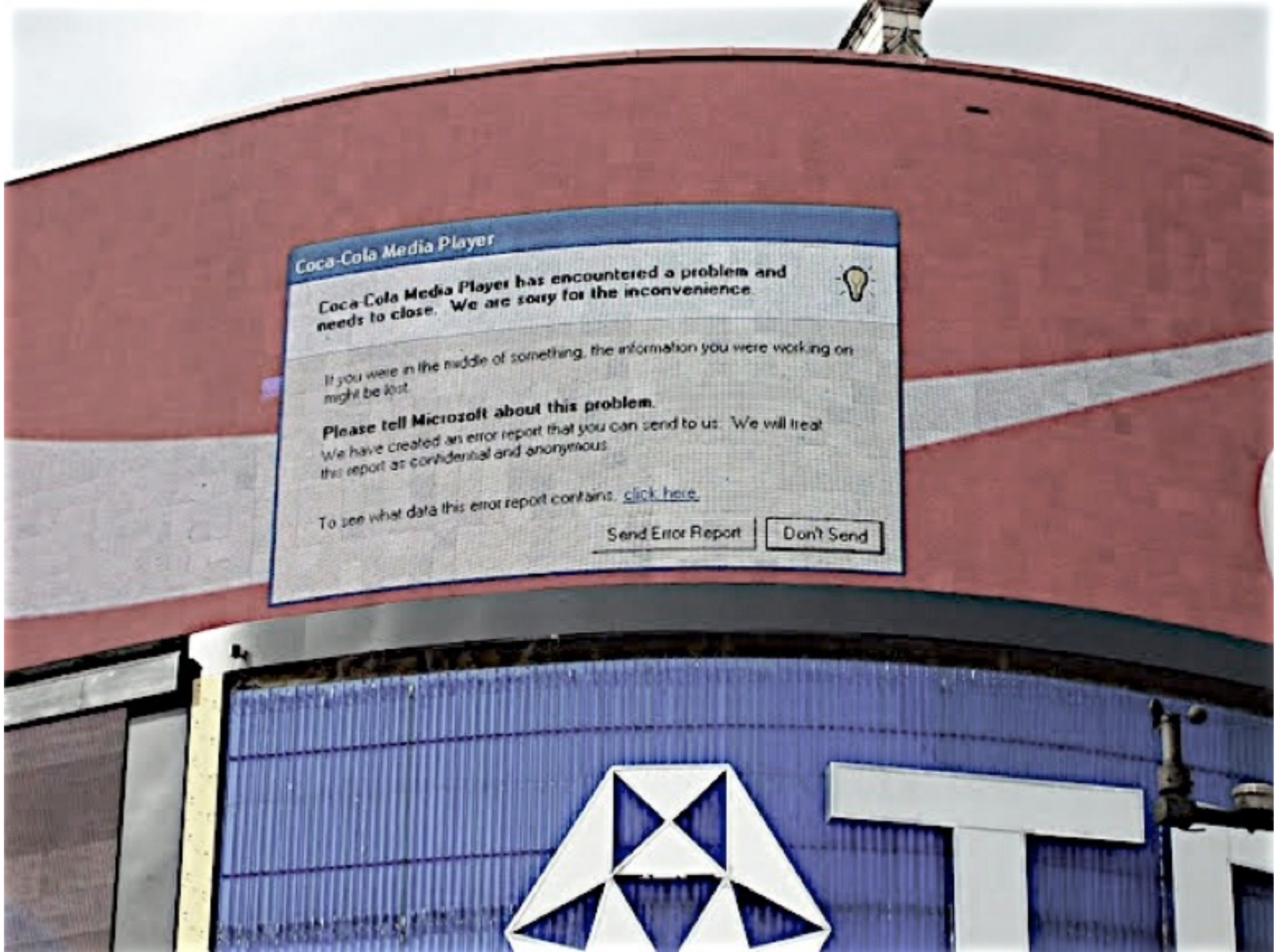
Duke of Wellington
Arthur Wellesley
1769-1852, British Statesman, Military Leader



What is Engineering?

„Engineering is the application of mathematics, empire, and scientific, economic, social, and practical knowledge for innovation, development, production, maintenance, and improvement of structures, machines, tools, systems, components, materials and processes.“

What happens when engineering fails?


Coca-Cola Media Player
Coca-Cola Media Player has encountered a problem and needs to close. We are sorry for the inconvenience.
If you were in the middle of something, the information you were working on might be lost.
Please tell Microsoft about this problem.
We have created an error report that you can send to us. We will treat this report as confidential and anonymous.
To see what data this error report contains, click here.
Send Error Report
Don't Send


Windows
An exception 06 has occured at 0028:C11B3ADC in VxD DiskTSD(03) +
00001660. This was called from 0028:C11B40C8 in VxD voltrack(04) +
00000000. It may be possible to continue normally.
* Press any key to attempt to continue.
* Press CTRL+ALT+RESET to restart your computer. You will
lose any unsaved information in all applications.
Press any key to continue

How Vulnerable Is Your Car to Cyber Attack?

As cars barrel toward full electronic control, are they vulnerable to cyber attack?

By Glenn Derene

Displaying an arbitrary message and a false speedometer reading on the Driver Information Center. Note that the car is in Park.

June 21, 2010 1:00 PM

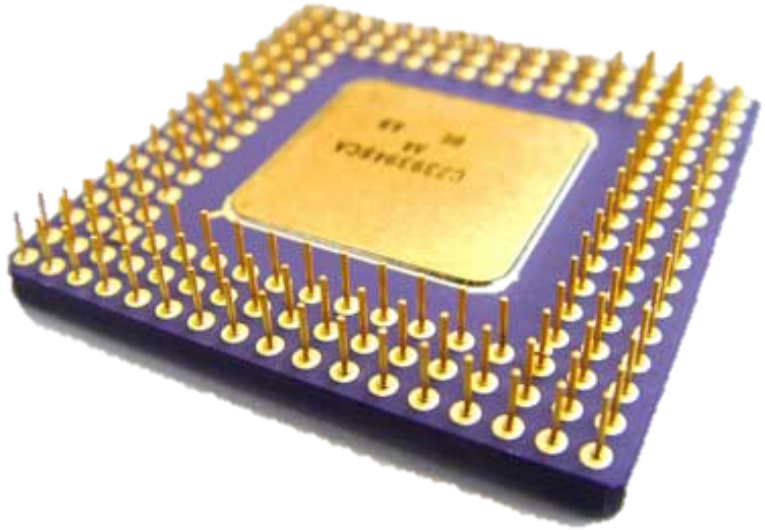
Last November, on a closed airport runway north of Seattle, Wash., a team of researchers from University of Washington and University of California–San Diego performed an ominous experiment on a late-model sedan. With a chase car driving on a parallel runway, they sped the test vehicle up to 40 mph, then turned off the brakes—via Wi-Fi. "Even though we knew what was going to happen, it's a very unsettling feeling to have a loss of control," says Alexei Czeskis, the researcher who was driving the test car. "You get full resistance from the brake pedal, but no matter how hard you press, nothing happens."

The test sedan was rigged up with a laptop hooked into its OBD II diagnostic port. On the computer was a custom-coded application, called CarShark, that analyzes and rewrites automobile software. That laptop was linked via a wireless connection to another laptop in the chase car. In addition to temporarily rendering the test car brakeless, the setup also allowed the research team to remotely turn off all the vehicle's lights (including the headlights and brake lights), turn on the windshield wipers, honk the horn, pop the

Popular Mechanics, 2010

- **3-5 Million ICs were shipped out but had a fault causing the division of numbers to fail in some cases.**

- **Causing damage of $ 475 Million**



Source: http://www.wired.com/software/coolapps/news/2005/11/69355?currentPage=2

- Fault when translating numbers from one format into another
- overpowers the rocket's engines
- causes the rocket to disintegrate 40 seconds after launch.
- Causing damage of $ 370 Million

Source: http://www.wired.com/software/coolapps/news/2005/11/69355?currentPage=2

What is Software Engineering?

- *"the systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software"* – IEEE Systems and software engineering – Vocabulary

- *"an engineering discipline that is concerned with all aspects of software production"* – Ian Sommerville

Subdisciplines of Software Engineering

- **Requirements engineering**: The elicitation, analysis, specification, and validation of requirements for software.
- **Software design**: The process of defining the architecture, components, interfaces, and other characteristics of a system or component.
- **Software construction**: The detailed creation of working, meaningful software through a combination of programming (aka coding), verification, unit testing, integration testing, and debugging.
- **Software testing**: An empirical, technical investigation conducted to provide stakeholders with information about the quality of the product or service under test.
- **Software maintenance**: The totality of activities required to provide cost-effective support to software.

Subdisciplines of Software Engineering

- **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.
- **Software development process**: The definition, implementation, assessment, measurement, management, change, and improvement of the software life cycle process itself.
- **Software engineering models and methods**: impose structure on software engineering with the goal of making that activity systematic, repeatable, and ultimately more success-oriented
- **Software quality**: for assuring that the software meets pre-defined quality goals and measures

Software Engineering Processes

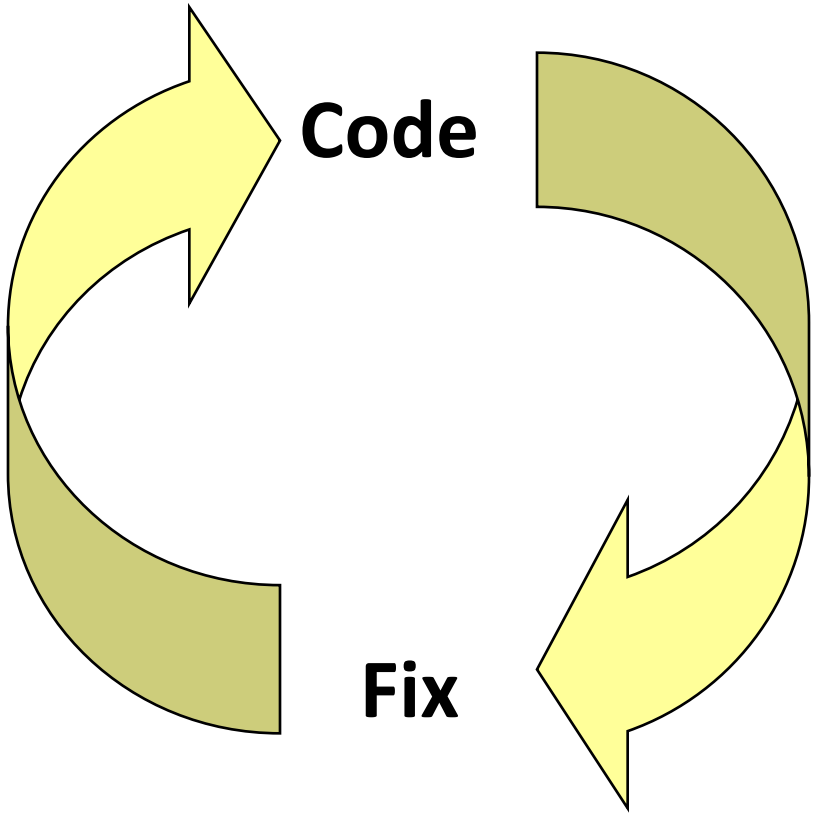
SE Process – Some remarks

- At the beginning of programming: 1 developer writing small programs
- Over time:
  - Development of huge programs (up to 100 Million LOC and more!)
  - Team of developers
  - Huge number of users (100 Millions and more)
  - Quality assurance becomes more and more important

$\Rightarrow$ There is a need for a development process!

Simplest process Code-And-Fix

- Code phase:
  - Programming
- Fix phase:
  - Corrections and
  - Extensions
- Ad-Hoc process model



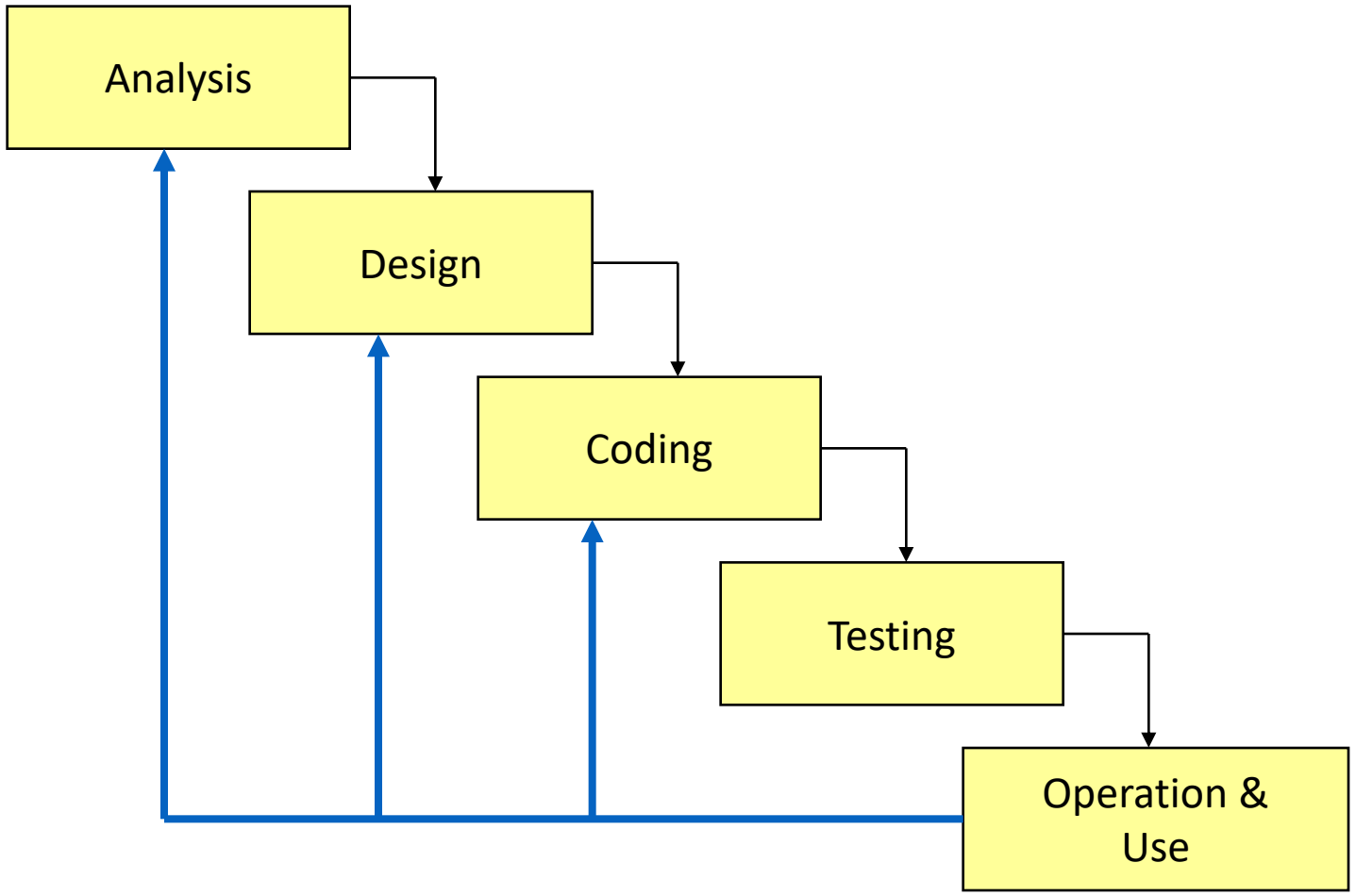
Analysis of Code-And-Fix

- Some phases are missing!
  - Analysis
  - Design
  - Verification & validation
- There is no time to think about relationships and other issues
- Source code still error-prone and becomes hard to maintain
- Ripple effects can only be detected when a failure accurs (i.e., a fault becomes visible)

The waterfall process model

- Add more details to the SE process
  - Decide what to do
  - Decide how to do it
  - Execution of phases
  - Verification & Validation
  - Use of software
- Structure in form of a cascade
- Outputs of one phase are the inputs of the next following phase

Waterfall process model



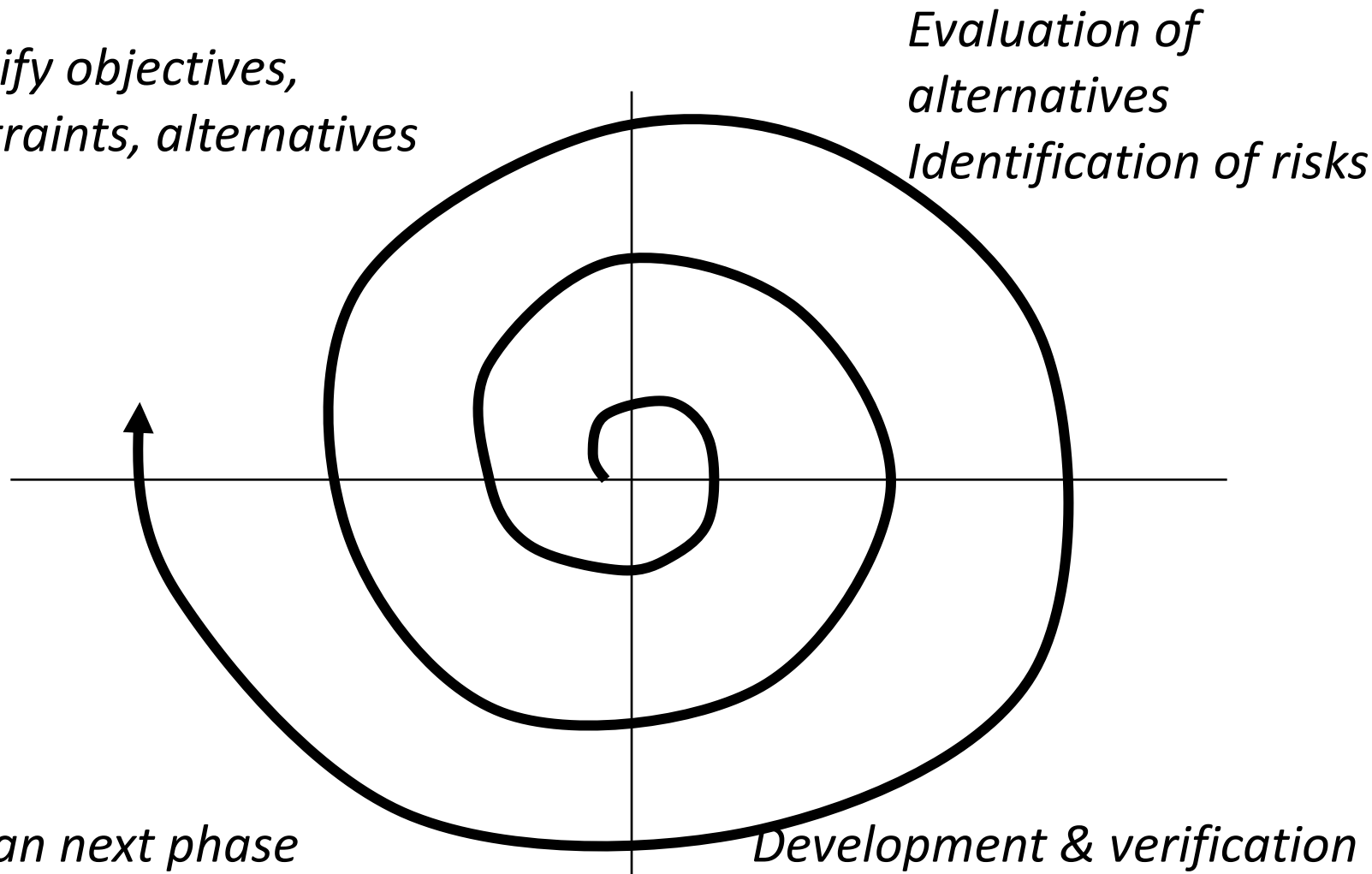
Analysis of the waterfall process model

- Faults occuring in preliminary phases but detected in later phases are very expensive!
- Hidden assumption
  „All results of one phase are correct and complete“
  is not realistic !
- Evolutionary character of software is not repected in the right manner!

Spiral process model

- Is based on 4 parts that are executed more often iteratively:
  1. Analysis of aims and goals, the requirements, and alternatives
  2. Evaluation of alternatives regarding their risks
  3. Next iteration of the program is going to be developed and also tested (verification & validation step).
  4. The next iteration is planned

Spiral process model



Analysis of spiral process model

- High flexibility
- Framework for other processes!
- *Example*: Spiral process model together with Code-And-Fix
- Risk driven!

Analysis Spiral process model (cont.)

- Challenges:
  - There is no assignment between requirements and audit/accountability
    - *Example*: Because of temporal requirements there might be no risk analysis
  - Risk analysis leads of to develop the easier parts first (instead of the most important ones)!

Agile processes

- Example processes
  - Extreme programming
  - SCRUM
  - Kanbaan

- Objectives:
  - ***Individuals and Interactions*** *over processes and tools*
  - ***Working Software*** *over comprehensive documentation*
  - ***Customer Collaboration*** *over contract negotiation*
  - ***Responding to Change*** *over following a plan*

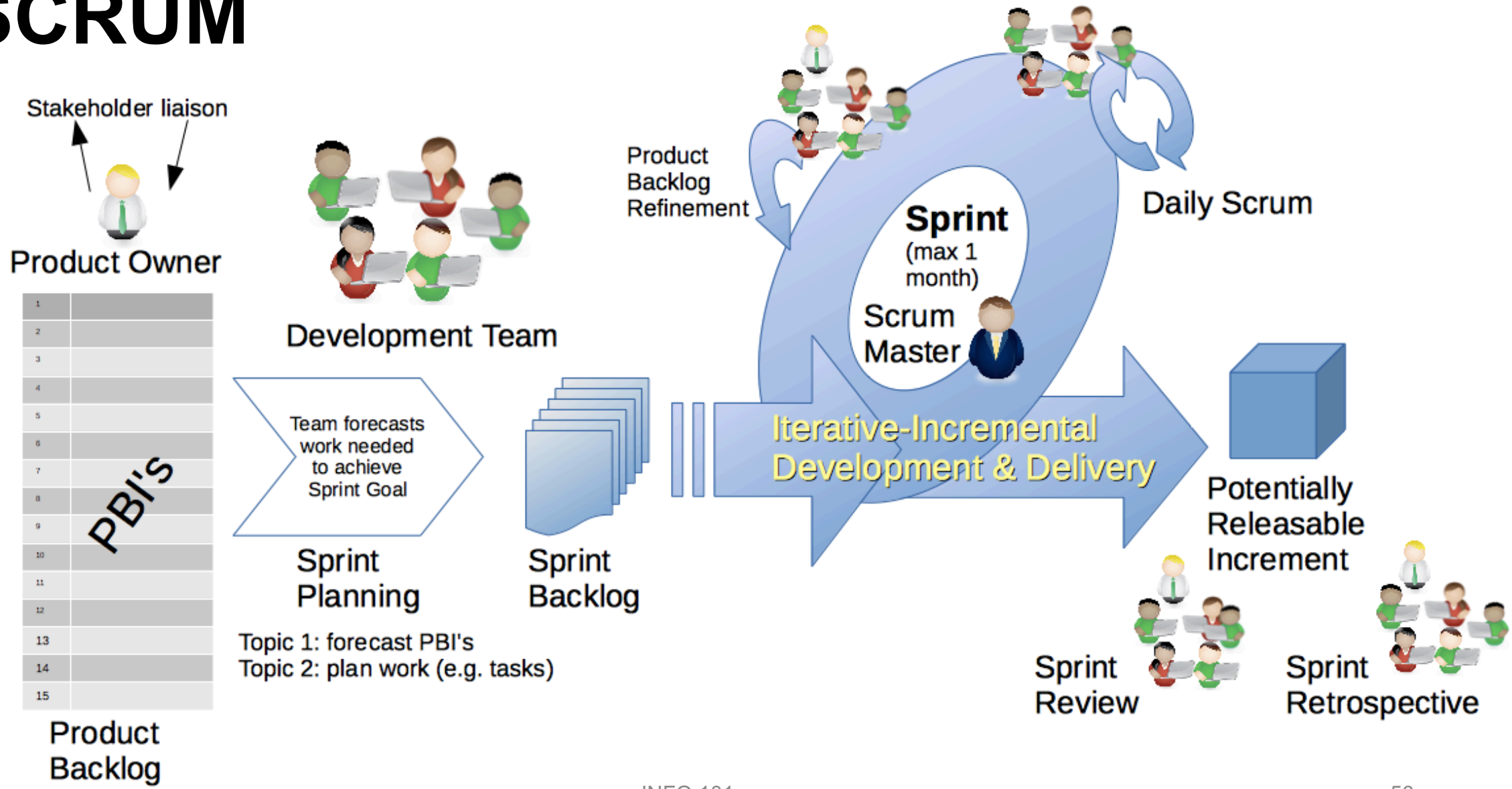
Manifesto for Agile Software Development has the twelve principles

- Customer satisfaction by early and **continuous delivery of valuable software**
- **Welcome changing requirements**, even in late development
- **Working software is delivered frequently** (weeks rather than months)
- Close, **daily cooperation between business people and developers**
- Projects are built around **motivated individuals, who should be trusted**
- **Face-to-face conversation** is the best form of communication (co-location)
- **Working software** is the primary **measure of progress**
- **Sustainable development**, able to maintain a constant pace
- Continuous attention to **technical excellence and good design**
- **Simplicity**—the art of maximizing the amount of work not done—is essential
- Best architectures, requirements, and designs emerge from **self-organizing teams**
- Regularly, the **team reflects** on **how to become more effective**, and adjusts accordingly

Some basic principles

- Iterative, incremental and evolutionary process using short delivery cycles
  - Sprints (2-4 weeks)
  - Daily builds (working software every day)
- Quality focus
  - Continuous integration: All changes are applied directly and also tested
  - Automated unit testing
- Just enough documentation

SCRUM


Stakeholder liaison
Product Owner
Development Team
Product Backlog Refinement
Sprint
(max 1 month)
Scrum Master
Daily Scrum
PBI's
Team forecasts work needed to achieve Sprint Goal
Sprint Planning
Sprint Backlog
Iterative-Incremental Development & Delivery
Potentially Releasable Increment
Topic 1: forecast PBI's
Topic 2: plan work (e.g. tasks)
Product Backlog
Sprint Review
Sprint Retrospective

Analysis of agile processes

- Often applied in practice of increasing importance
- Programmers often like agile processes
- Working software is available as early as possible
- Lack of documentation face problems for maintenance
- May lead to a software architecture that can hardly be maintained
- Innovation can hardly be introduced in the process (because the focus is on the product)

SE process models – Some remarks

- SE processes are important for
  - Developing high quality software
  - In time development
  - Know your costs
- The operational use of SE processes is maybe more important for practice!
  ⇒ Often processes are officially used but not used in practice!
- How good are processes used?
  ⇒ **Capability Maturity Model**

Capability Maturity Model

1. Initial: The used SE process is ad-hoc. Only some parts are defined and used. The result can hardly be preticted both in quality and costs.
2. Repeatable: Base process are installed and used. Costs, the process execution etc. are tracked. Success can be repeated in case of similar projects.
3. Defined: Processe are standardized and documented. There are standard processes used in all projects.

Capability Maturity Model

4. Managed: Detailed measures are taken for the process and the product quality. The process is understood quantatively and control of process execution is established.
5. Optimising: The quantitative feedback during development and after finalizing a development is evaluated for the purpose of continous process improvement!