Hochschule Esslingen
University of Applied Sciences

MSc Industrial Automation - System Design

Prof. Dr.-Ing. Werner Zimmermann

I. System Development Process

• Definition of a system	
• Decision making:	– definition of goals – Eisenhower principle – weighted criteria assessment
• Planning:	– activity lists – time schedule
• Development process	– top down, bottom up – partitioning and interfaces
• Phase models	– actions, results, milestones
• Requirements analysis	

II. Analytical System Design

• Modeling dynamic systems	– state variables – describing equations – block diagrams
• State space control	– state space description – control structure – pole placement design
• Observers	– observer structure – pole placement design
• Optimization of control systems	
• Identification of control plants	
• Neuronal Networks	– application in control systems

Module System Development Process

Contents

Index

Module System Development Process

A. Overview

1. Technical systems

A car as a technical system:

A system

- consists of **components**

  e.g. engine, brakes, ...

- performs **functions**

  e.g. accelerating, braking, ...

- has **external interfaces** (to the outside world/environment)

  e.g. accel. and brake pedal, tires, ...

  and **internal interfaces** (between the components)

  e.g. accelerator pedal - engine

Most systems

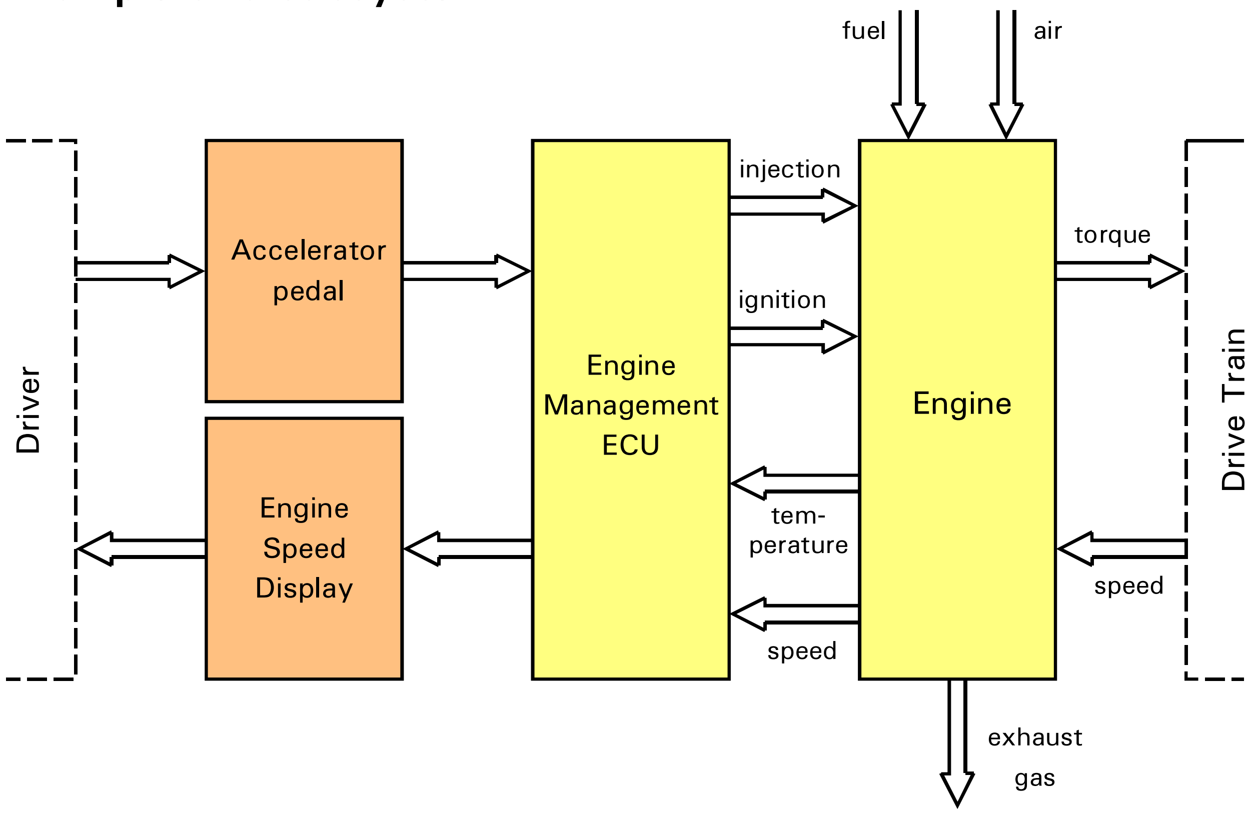
- are components of **super-systems**, i.e. higher level system

  e.g. super-systems of a car:
  - a car is a component of the system 'traffic'
  - 'traffic' is a component of the system 'city'
  - . . .

- can be subdivided in **subsystems**, i.e. lower level systems

  e.g. subsystems of a car:
  - engine and engine management system
  - brakes and brake management system (ABS/ASR)
  - . . .

Example of a subsystem



System border

- defines what is considered to be inside and outside of the system
- can be arbitrarily chosen by the system engineer depending on what is considered to be of interest

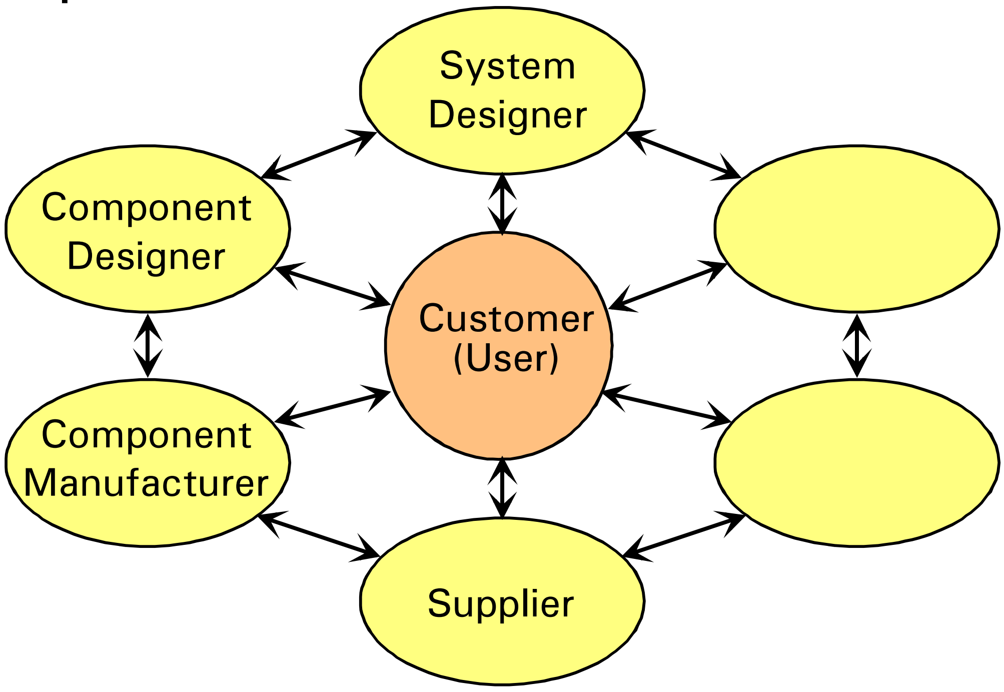
2. Designing a system means

- establishing **project goals**
- **defining the functions**, which the system shall perform
- and “**border conditions**” like size, cost, ... User's view

- -

- **planing** the development project Developer's view
- **checking and refining the requirements**
- **partitioning** the system (system concept/architecture), i.e.
  - identify, which components are needed and
  - assign functions to components
  - defining the internal interfaces between the components
- **developing** or purchasing the components

Involved parties



3. Development project types

Technical products can be divided in 3 major categories. Depending on the category, customer involvement during the project and intensity of the project phases differ very much:

- **Multiple customers - high volume products** ("mass product")

  e.g. TV sets, mobile phones, office standard software

– At the end of the development process the **product "waits" for customers.** The customer is not involved in the development process, a marketing department within the company defines technical features and target price.

– Analysis of **competitor products** is crucial.

– New products are **feature enhanced** old **products**.

– **Price**, **public relation/marketing** and **time to market** are often more **important** than technical features.

- **Single customer - high volume products**

  e.g. automotive electronics

– Product **design** is mostly **customer specific**. The **customer** is **heavily involved** in planning, requirements analysis, design decisions and testing. $\rightarrow$ Development partnership

– In many cases the product is an add-in to the customer's own product. In these cases the **development** process must be **synchronized to the customer**'s own development.

– **Further development** activities **during operation/maintenance** phase to prolong production lifetime and for continued parts cost reduction.

- **Single customer - low volume product**

  e.g. steel rolling mill, automobile production equipment

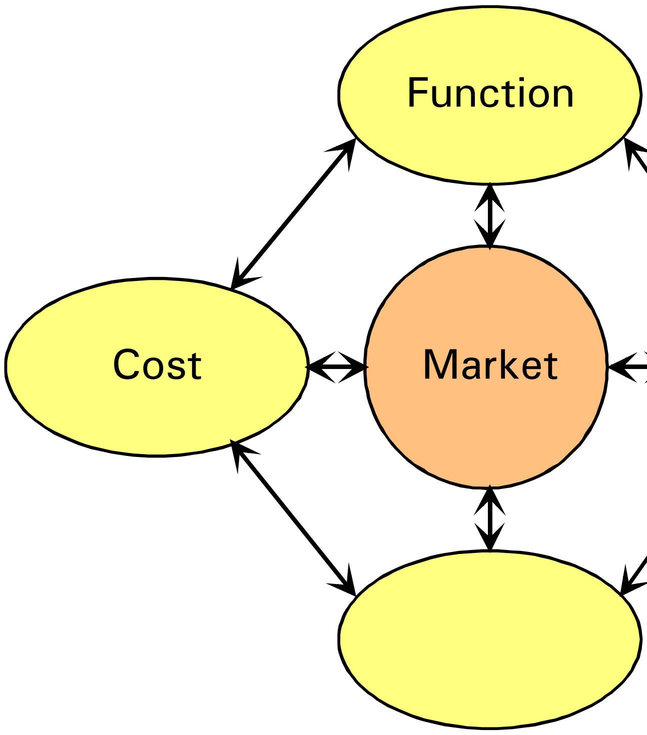
– Design is completely customer specific. $\rightarrow$ Engin. service

– Support during operation and maintenance of the "product" are as important as the "product" development.

A comparison:

	Multiple customers high volume products	**Single customer high volume products**	**Single customer low volume product**
Customer involvement	none	very important + + +	
Technical features	less important	very important	
Parts cost	very important + + +		not important – – –
Development time	short months	medium … long 1 … 5 years	long 2 … 5 years
Development cost	low	very high	very important + + +
Time to market	very important + + +	Important	not important
Operation/ maintenance	no	product care cost reduction	very important
Product marketing	very important	not important	
Production lifetime	Short months	Medium 2 … 5 years	Very long (incl. maintenance) 5 … 10 years

4. Development project goals must be defined in terms of



Goals must be	**e.g. PC development project**
• **precise**	– low cost PC + total cost below $500
• **measurable**	– optimum performance + performance under defined test conditions, e.g. benchmark test
• **achievable**	– gain 100% market share + increase our sales volume by 25%
• **consistent**	– use newest Intel CPU + use CPU which costs < $200

Goals may be conflicting, especially between involved parties.

Example: Study MSc ITAE in Esslingen

Your goals:
1. Graduate with best grade in all subjects
2. Graduate in the shortest possible time
3. Earn $ 3000,- per month with student jobs

Will you be successful?

Most likely not!

Discussing the goals:

↓ Criteria Alternatives →

	Goal 1	Goal 2	Goal 3
precise?	Yes	No → specify time or date	Yes
measurable?	Yes	No → time or date: yes	Yes
achievable?	Only if you are a genius !?	Yes	No, students may only work <40hrs/month
consistent?		No, need more time to work	

Dealing with conflicting goals: **Assign Priorities**

e.g.

A (highest priority)	**must** be a achieved → a 'killer' feature
B	**shall** be achieved
C	**may** be achieved → a 'nice to have' feature

How to set priorities?

- **Eisenhower's principle**

Don't confuse: **important – urgent**

	very important	less important
very urgent	A • start at once • need to do it yourself	B • look for help • let somebody do for you (delegate)
less urgent	C • keep for later • schedule in your activities list	W • forget about it ! • put into the waste paper basket

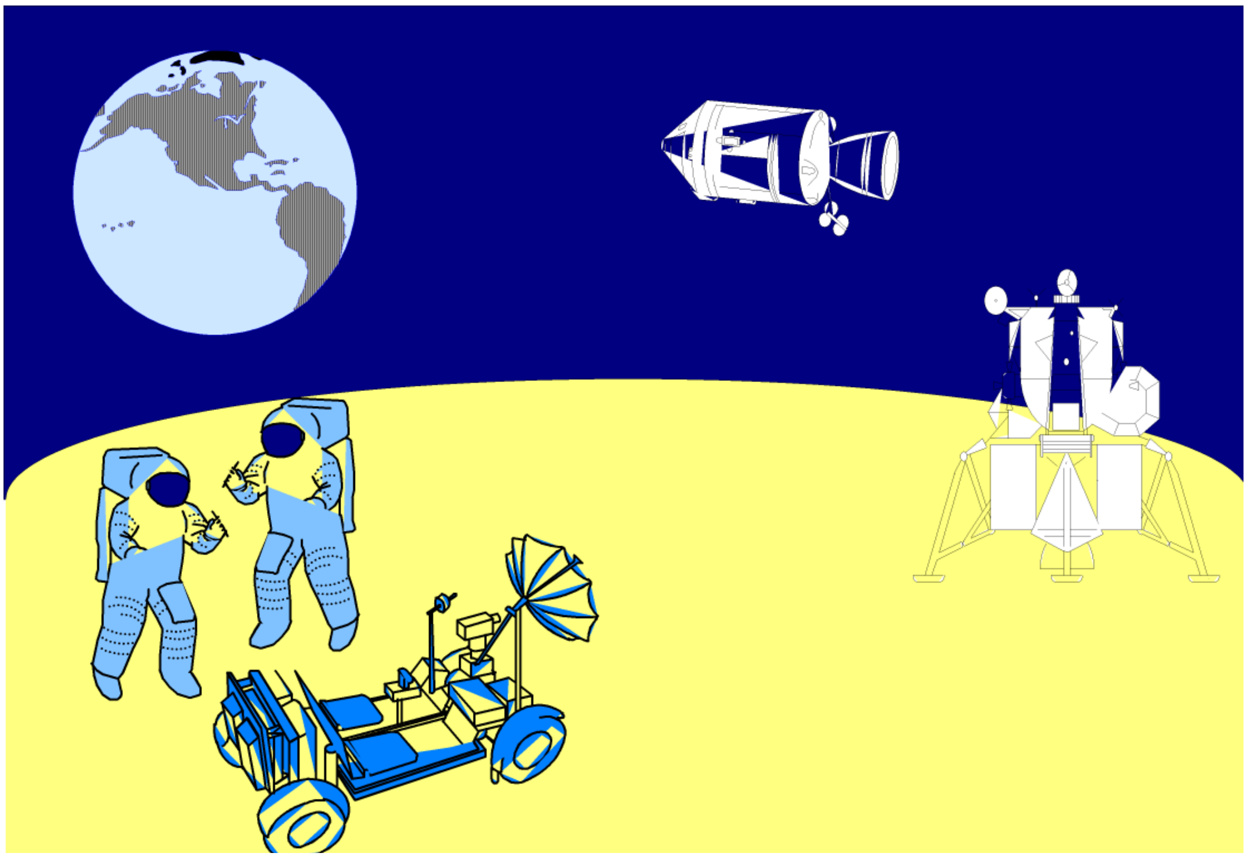
- **Pareto's rule: The 80% – 20% principle**

80% of the total result can be achieved with 20% of the total effort. The remaining 20% of the functions take more than 80% of the effort!

- Identify the 'must' functions $\rightarrow$ A
- The 'rest' goes into the B and C category
- Use incremental development:
  - Implement the must functions and create a working prototype with limited functionality
  - Add functions and features incrementally according to B and C categories
  - Finally optimize

B. Decision Making and Planning

1. The NASA game



Imagine the following situation:

- You are in a team of 2 astronauts, who have landed on the **bright side of the moon.**
- You are on an excursion with your lunar vehicle some **100 km away from your lander module** (the rocket, which will bring you back to your mother ship in the moon orbit).
- Unfortunately the lunar vehicle has broken down and you have to **walk back to your lander module.**
- For such cases you have an **emergency equipment**, consisting of the **12 items**.
- However, you cannot carry all these items with you and most likely during your march you will have to leave back items from time to time. So you have to **decide**, **which items are more important** for your survival than others.

The items in your emergency kit are (unordered list) :

- 90-110 MHz sender/receiver
- Parachute
- Matches
- First aid kit
- Stellar map (moon and stars)
- 20 m rope
- Signal pistol with flares
- Portable heater
- 20 l water
- Compass
- Two 5o kg oxygen tanks
- Food concentrate

These items shall be assigned **unique priorities from 1** (most important) **to 12** (least important). No two items must have the same priority.

The assignment is done in two rounds:

a) Each student delivers his/her individual solution.

b) Students discuss and a agree on a team solution with 4...5 students being in a team.

a) Individual solution:

Student name: ______________________________

Items in my emergency kit:

Item	*priority *1*
90-110 MHz sender/receiver	
Parachute	
Matches	
First aid kit	
Stellar map (moon and stars)	
20 m rope	
Signal pistol with flares	
Portable heater	
20 l water	
Compass	
Two 5o kg oxygen tanks	
Food concentrate	

*1

Please **assign priorities from 1** (most important) **to 12** (least important). No two items must have the same priority.

b) Team solution:

Team members: ______________________________

____

____

____

Items in our emergency kit:

Item	*priority *1*
90-110 MHz sender/receiver	
Parachute	
Matches	
First aid kit	
Stellar map (moon and stars)	
20 m rope	
Signal pistol with flares	
Portable heater	
20 l water	
Compass	
Two 5o kg oxygen tanks	
Food concentrate	

*1

Please **assign priorities from 1** (most important) **to 12** (least important). No two items must have the same priority.

Observations:

- Decisions can not always be made based on pure facts and real knowledge → deal with uncertainty
- Decisions in groups can be influenced by communication skills of group members → deal with the "human factor"

2. A formalized, semi-objective way of decision making: Weighted criteria assessment

- **Define** the **alternatives**
- **Formulate** the **criteria** to assess the alternatives
  Use positive criteria rather than negative ones (if possible)
- **Assess** the **importance of** the **criteria** ("priority scale")
  Use a 'weight' scale, e.g. 10 for 'most important', 1 for 'least important'
- **Assess** the **alternatives** criterion by criterion
  Use a point scale, e.g. 10 for 'best', 0 for 'worst'
- **Multiply** your weights with your points **and sum up** for each **alternative**

An example situation:

- You are a FHTE student, looking forward to the 2 months summer break.
- Next semester you will start working on your master thesis.
- You have an old car in very bad condition. A new car would cost $1500 minimum.
- You have a friend, living in Madrid/Spain. Visiting your friend would cost $800 minimum.
- You have $1200.

What to do: Work and buy a new car or visit your friend?

Hints for using this method:

- Try to spread the weight and the assessment scale to better differentiate as much as possible.
- If sums are close to each other, the decision still is unclear. Review your weights or try to find more criteria.
- Remove those criteria, which nearly have the same meaning (e.g. “gain working experience” and “gain experience in foreign countries”) and try to reformulate one common criterion, including both goals (e.g. “gain experience which is helpful for your future job”).

  → By having to many criteria with nearly identical meaning the decision process can be easily manipulated in favor of one alternative.
- Try to find compromise alternatives.
- Don't trust blindly! Critically review your results. In case of unexpected results, check your criteria and weights.

It is the discussion process, forcing you to think about, express and understand, what you and the others in the team really want or feel, which makes this method valid, not the naked result values.

			Alternative 1 Work and buy a new car		Alternative 2 Visit your friend in Madrid		Alternative 3	
#	Criterion	Weight	Points	Points x Weight	Points	Points x Weight	Points	Points x Weight
	Ensure mobility	**6**	**10**	**60**	**0**	**0**		
	Recreate from student life stress	**1**	**1**	**1**	**10**	**10**		
	Relationship to your friend	**4**	**0**	**0**	**10**	**40**		
	Gain experience which is helpful in your future job	**10**	**8**	**80**	**5**	**50**		
			Sum:	**141**	**Sum:**	**100**	**Sum:**	

			Alternative 1		**Alternative 2**		**Alternative 3**	
#	Criterion	Weight	Points	Points x Weight	Points	Points x Weight	Points	Points x Weight
			Sum:		**Sum:**		**Sum:**	

3. Planning

... is the key to successful projects. Nearly all every day life activities include some amount of planning.

Examples:

- traveling:
  - Where to go?
  - Which routes to take to get there?
  - When does the bus/train/plane go?
  - Hotel/camp ground reservation made?
  - Passport valid, visa needed?
  - Credit card, foreign currency, traveler cheques available?
  - How many and which clothes needed?
  - . . .
- buying food:
  - What do you want to eat/drink?
  - Contents of refrigerator checked?
  - In which shops to buy?
  - "Transportation" capacity sufficient?
  - Enough money available?
  - . . .
- . . .

Planning consists of 3 tasks:

a) Collecting the data basis for planning

b) Scheduling activities (time table = "making the plan")

c) Continuos controlling and correcting the plan ("feedback")

3.1 Planning - step by step

a) **Collecting the data basis for planning**

1. Define **activities** - **what must be done ?**
   - Write down all activities, which must be performed during the project $\rightarrow$ **activity list**
   - Specify the expected result of each activity. Define, how to check, whether an activity was carried out successful.
   - Start with a brainstorming approach, i.e. write down activities without any order as they come into your mind.
   - Then try to group the activities, e.g. by assigning them to the different project phases (see chap. C).
2. Define **responsibilities** - **who does it ?**
   - For each activity write down, who has to perform it ("human resources"), i.e. who is responsible, that the desired results are available in time. Always assign humans ("personal responsibility") not organizations like companies, departments, ... !
3. Define **prerequisites** (interdependencies) - **what is needed before the activity can begin ?**
   - For each activity think about, which results from other activities will be needed, before this activity can begin (e.g. PCB design does only make sense, when the circuit design is completed and circuit diagram and parts list are available).
4. Define technical **resources** - **what is needed during the activity ?**
   - E.g. EMC test cell for EMC tests, prototype vehicle, ...
5. Define **needed/available time** - **how long does the activity take ?**

- For each activity estimate, how long it will take to carry out the activity.
- In some cases the available time for an activity (or the whole project) is forced by conditions outside of your control (e.g. road winter testing for vehicles in Europe is limited to December to March, your master thesis must be finished within 6 months, ...). This will define the amount of resources needed, to carry out all activities in the available time or limit the number of activities, which can be carried out.

b) **Scheduling the activities**

6. Define **the sequence** **- when will it be done ?**

- The interdependencies of the activities (step 3) and the human (step 2) and technical resources (step 4) define a logical sequence of the activities.
- Activities, which do not conflict with respect to resources and interdependencies can be paralleled to reduce the overall time.
- Together with the needed or available time (step 5) for each activity a → **time schedule** (the "plan") can be established. Time schedules typically are presented as GANTT or PERT diagram.

c) **Continuos controlling and correcting the plan ("Feedback")**

- The begin and end of all activities must be monitored and the results checked. If the reality and the plan differ (the usual case), the plan must be corrected.
- At the beginning of a project, the activities may be rather coarse, summarizing rather than going into detail. During the project details will be worked out step by step. Typically at the end of each phase, the next phase(s) are planned in more detail.

3.2 A planning example

Establish a coarse plan for the development of a mobile phone.

Note:
To simplify the plan, activities for

- market analysis,
- marketing campaign and introduction
- production preparation
- documentation and certification
- . . .

will not be considered here.

Remarks:

- Activity lists and schedules are partially redundant. Activity lists are better suited for details. Schedules are better suited to show the sequence of the activities.
- Often various levels of detail are necessary, e.g. for management a schedule with major phases is sufficient, whereas for engineering staff activities must be broken down to details.
- To reduce redundancy, allow various views on the data and easily adapt the plan to any changes during the project, **project management software** like MS Project is available. Other tools like Outlook or PDA software may also offer partial planning support.

However:
Planning is still an 'engineering' task and cannot be done automatically!

Activity list

#	What?	Who?	How long?	When?		Done?
	Activity - Prerequisites (PR) - resources (RS) - results (R)	Responsibility	Duration	Begin	End	Check
1	Requirements analysis PR: Marketing analysis, R: Req.Doc	Marketing Prod.Planning	2w	See Schedule		
2	Concept phase PR: 1, R: System Architecture and Spec. Doc	HW, SW, ME concept team	2w			
3	Mechanical design PR: 2, R: Case prototype	ME dept.	3w			
4	Hardware design PR:2+(3), R: PCB prototype	HW dept.	6w			
5	Mechanical + HW integration PR: 3 + 4, R: Working prototype HW	HW, ME	1w			
6	Software design PR: 2, RS: Breadboard HW, R: SW prototype (?)	SW dept.	8w			
7	Hardware + software integration PR: 4+6, Working prototype	HW, SW	2w			
8	Prototype test PR: 5+7, R: Test documentation	Test Team	4w			
9	Redesign/Retest (details like 4...9) PR: 9, R: Production ready product	HW, SW, ME	4w			

Schedule

#	What?	Who?	When?												(scale is CW=calender weeks)												
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
1	Req. analysis	MG, PP																										
2	Concept phase	Concept T.																										
2.1	Concept approval	Chief Eng.																										
	Implementation																											
3	Mechanical design	ME																										
4	Hardware design	HW																										
5	ME/HW integration	HW, ME																										
6	Software design	SW																										
7	HW/SW integration	HW, SW																										
7.1	Prototype release	Chief Eng.																										
8	Prototype test	Test T.																										
8.1	Test evaluation	Chief Eng.																										
9	Redesign/Retest	All																										
9.1	Release for Prod.	Chief Eng.																										

Legend: △ Begin of activity ▽ End of activity ◇ Milestone ▲▼◆ Done

Activity list

#	What?	Who?	How long?	When?		Done?
	Activity - Prerequisites (PR) - resources (RS) - results (R)	Responsibility	Duration	Begin	End	Check

Schedule

#	What?	Who?	When? (scale is)																									
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

Legend: △ Begin of activity ▽ End of activity ◇ Milestone ▲▼◆ Done

3.3 Keys to successful planning

1. **Agreement with all involved parties**

- If others are involved in your project to carry out activities, deliver input for your project or use your results, let them specify their own activities and the time needed for them.
- No project will succeed, if not all involved parties agree to their part of the activity list and time schedule.

2. **Don't overconstraint the plan**

- A plan is a **tradeoff between activities, resources and time**. Not all of these 3 items may be predefined:
- If activities and resources are predefined, the time is an output of the planning process rather than an input.
- If resources and time are predefined, the possible activities are an output, resulting in limited or canceled activities or activities carried out with limited quality.

3. **Don't use best case planning**

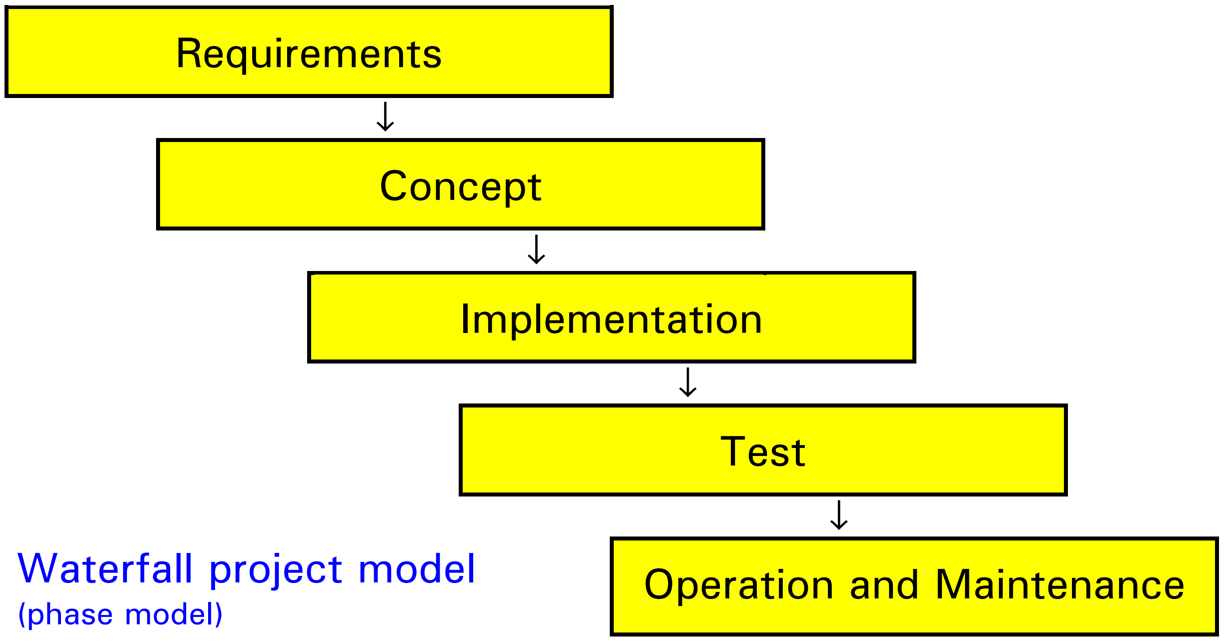
- Leave **room for the unexpected** in your plans! If anything can go wrong, it will ... (Murphy's law). "First time right" and "zero failure" are good goals, but hard to achieve.
- Include **milestones**, where to **check results** and include phases for correcting them ('**redesign**'). The time between the milestones shall be as short as possible (use 'sub-milestones'), to make the feedback loops small.
- **Sequence** activities **according to** their **priority**, put less important activities at the end! If the project runs out of time, cut or shorten these activities.

4. **Build up planning experience**

- Always review your projects to learn for the next project. What was good, what went wrong? Which was the actual time needed? Which partners were reliable?

C. Requirements Analysis and Project Phases

1. A simple project model



Waterfall project model
(phase model)

Each phase has

- **activities**

… to be performed to produce the phase's result

e.g. in the implementation phase:
circuit development, PCB development, …

- **results**

- "products", documenting the results of the phase
=input for next phase

e.g. at the end of the requirements phase:
hardware/software specification document

e.g. at the end of the implementation phase:
fully functional prototype

- **milestone(s)** or gate(s)
  - result(s) which must be achieved and (formally) be approved before proceeding to the next phase

    e.g. at the end of the test phase: release for production

In big projects each phase will be subdivided in subphases with parallel and sequential activities.

2. Requirements analysis

2.1 Example: Walkman AC Adapter

Original requirements specified by the customer:

- Develop an AC adapter for a walkman, to connect the walkman to the mains power supply. The adapter will be sold in Europe.

Refined requirements

- After the designer has interviewed the customer, a detailed requirements document can be written:

Requirements Document for FHTE-Walkman AC Adapter

1. Purpose
This paper describes an AC adapter for our FHTE-walkman, to connect the FHTE-walkman to the mains power supply.

2. Marketing Information
The product is to be sold as an option to those customers, who also buy our FHTE-walkman. Competition and their products' features were already described in the FHTE-walkman marketing study. However, the AC adapter will only be sold in Europe.

3. Function

3.1 Normal Operation
The AC adapter shall deliver a constant DC output voltage at a variable output current within the specified AC input voltage range. There shall be no user selectable or adjustable parameters.

3.2 Startup and Shutdown Procedure
The AC adapter shall start and stop operation, whenever the AC input connector is plugged into or removed from a mains socket. The load may or may not be connected during startup

and shutdown and may be connected or removed during normal operation.

3.3 Failure Behavior

The adapter's output shall be short circuit protected. The adapter may turn off its output voltage, if the adapter's internal temperature becomes too high. It must turn on again after cooling down, if the input plug is removed and reconnected.

3.4 Additional Functions

None

4. Technical Data

4.1 Electrical and Mechanical (see also 3.2) Interface

• Input	110 ... 230V ± 10% / AC 50 Hz 2pin Euro plug with 2.5m cable
• Output	9V DC ± 3%, 0 ... 500mA isolated from input 3.5mm socket

4.2 Case

• Material	non-conductive plastic
• Color	black or gray
• Insulation:	double insulated, comply with VDE 100
• Size	< 6 x 6 x 7 cm³
• Weight	< 250g
• Surface temperature	< 50°C

4.3 User Interface

• Switches and keys	none, turn on by plugging in the input cable
• Display, control lamps	LED (green), if output voltage is in specified range, brightness t.b.d.

4.4 Environmental Conditions

• Ambient temperature:	+10...30°C
• Air humidity:	10 ... 80%
• Sealing:	IP 44
• Electromagnetic compatibility:	CE
• Vibration and shock:	1g sine wave 1 ... 1000Hz must withstand a 1m fall on concrete floor

5. Commercial Data

• Projected sales volume:	10000 (1st 3 months) 100 000 (total in 3 years)
• Projected parts cost:	< $ 20
• Projected availability:	Preproduction samples May 2004
•	Start of Production Sep. 2004

2.2 Requirements for 'Requirement Documents'

- Requirement documents should be **written from a user**/customer's **view**, **defining, what the system**/product **should do**, not how it is implemented.

  During the **concept phase**, the requirements document is **converted into a 'Specification Document', which describes, how the system**/product **shall be implemented** together with a detailed project plan and a parts and development cost estimation.

Required contents:

- **Functions** of the system **in <u>all</u> operating modes.** Many customers forget to specify startup and shutdown procedures and the desired behavior in failure situations
- **Electrical and mechanical interfaces description**
- **User interface** description (how to operate the system)
- **Environmental conditions** (ambient temperature, sealing, vibration, EMC, ... for hardware; computer platform, operating system, disk/ram sizes, ... for software)
- A requirements document **should refer to or include** a short abstract of **a marketing study**, showing the main competitors, market share and competing products with key features and prices. Very often such a marketing study initializes the product development project.

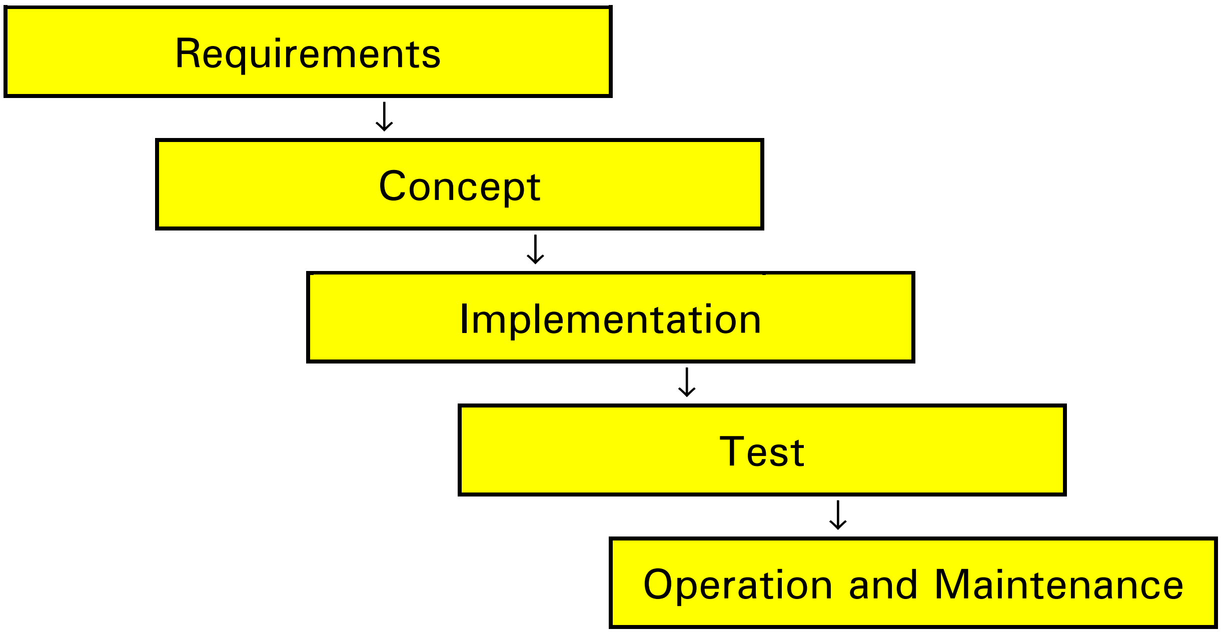
Style

- **Use data** rather than purely describing properties (minimum <u>and</u> maximum values or nominal values <u>and</u> tolerances if possible)
- **Use established standards** (e.g. IEC, ISO, ANSI, ...), where applicable
- **Include all items necessary for product development**. If these items cannot yet be defined, mark them as "t.b.d= to be defined later".

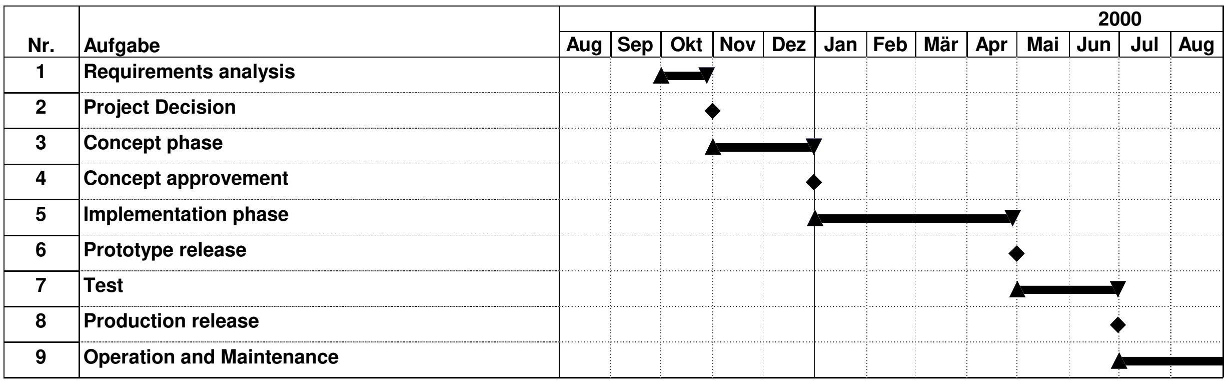
Intentionally left blank

3. Real world projects

3.1 Problems with the waterfall project model



The **waterfall model** is strictly **sequential** ...

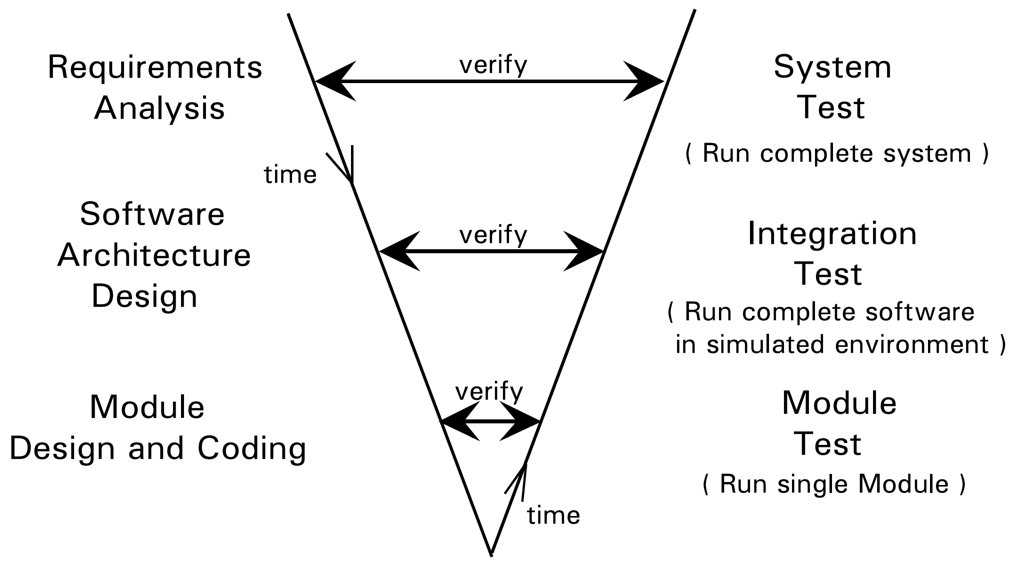


Nr.	Aufgabe
1	Requirements analysis
2	Project Decision
3	Concept phase
4	Concept approvement
5	Implementation phase
6	Prototype release
7	Test
8	Production release
9	Operation and Maintenance

. . . **real world** projects are not ! In real world projects

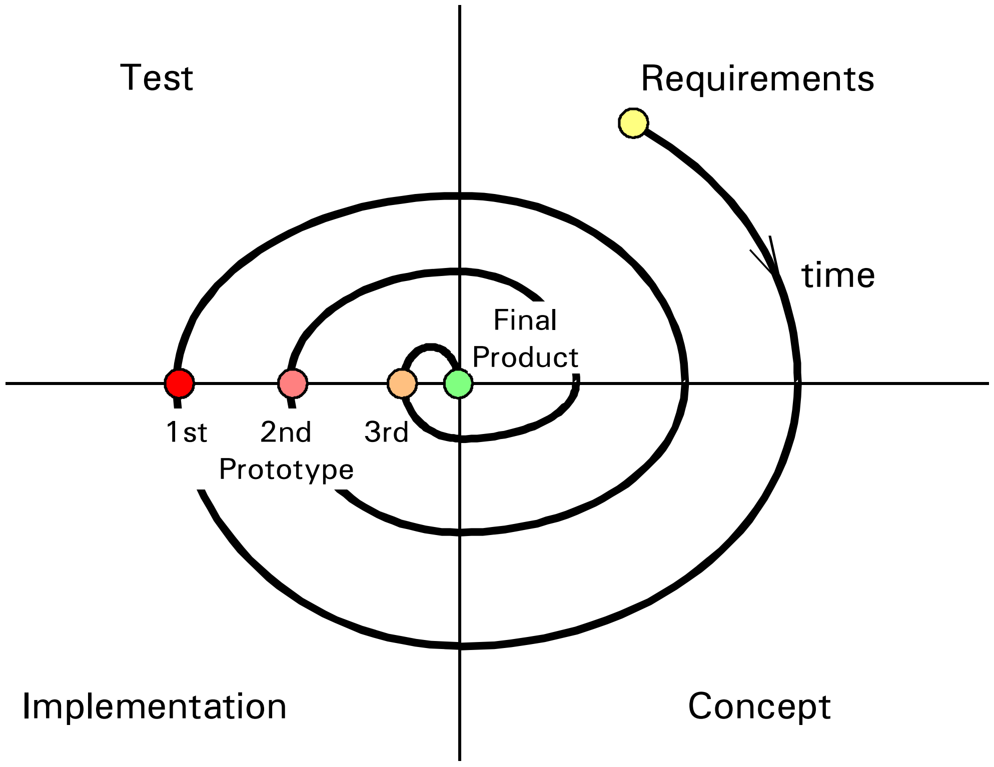
- due to missing information **not all details** for later phases **can be specified in early phases**
- due to technical problems **implementation tradeoffs have retrospective effects** on the concept or even on the requirements
- **time or resource constraints do not allow** a fully **sequential schedule**
- **customer or market demands** may **change** (the longer the project, the more this is likely)

- Long feedback loops V-model of software dev. BAD



Errors in the requirements analysis (first phase of the development process), causing a wrong system behavior, are often only found during the system test (at the end of the development process)

Spiral Model: Incremental design instead of "first time right"



- Incremental design: Plan to refine requirements and prototype function in a carefully planned step-by-step process

3.2 Improving the development process

- **Top-Down vs. Bottom-Up**

Top-down: Proceed from a global overview (goals, objects, block diagram level, interfaces,...) to the detailed level (algorithms, circuits, ...)

Bottom-up: Proceed from the detailed implementation level to the complete system

Example: Items to solve when designing a telephone network

TOP	How many users at which locations?
	How often and how long will the calls between the different locations be?
	...
MIDDLE	Wireless or wire-bound transmission?
	Optical or electrical?
	Analog or digital?
	...
	Class A or class A/B audio amplifier?
BOTTOM	RJ45 or TAE connector?

– Use a **top-down** approach **to get the big picture** fast (requirements, concept, major building blocks)

– **Identify the critical** technical **problems** ("job killers")

– Use a **bottom-up** approach **to find out** early, **if and how the critical technical problems can be solved**

$\rightarrow$ make a good concept and eliminate risks early

- **Rapid prototyping simulation and implementation tools**

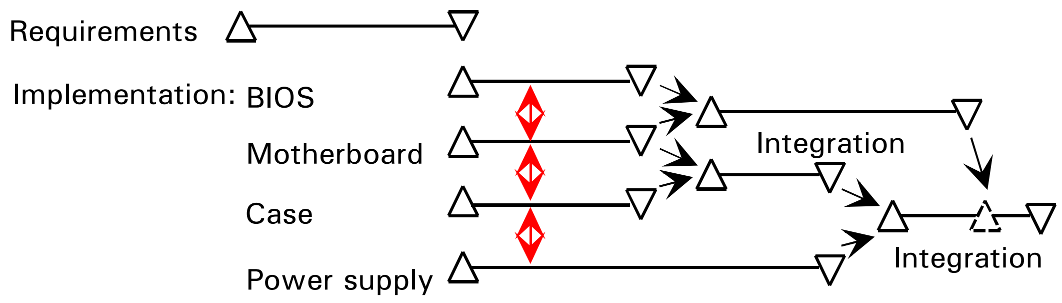
– Requirements and concept are verified by simulation rather than waiting until the implementation is completed.

– Prototypes are build by automatic generation tools (e.g. program code synthesis from UML models, circuit synthesis from VHDL/Verilog descriptions)

$\rightarrow$ reduce errors in requirements and implementation

- **Simultaneous engineering (Parallel design)**

– Split up the development between simultaneous working groups, e.g. PC development → HW/SW co-design



→ reduces overall development time

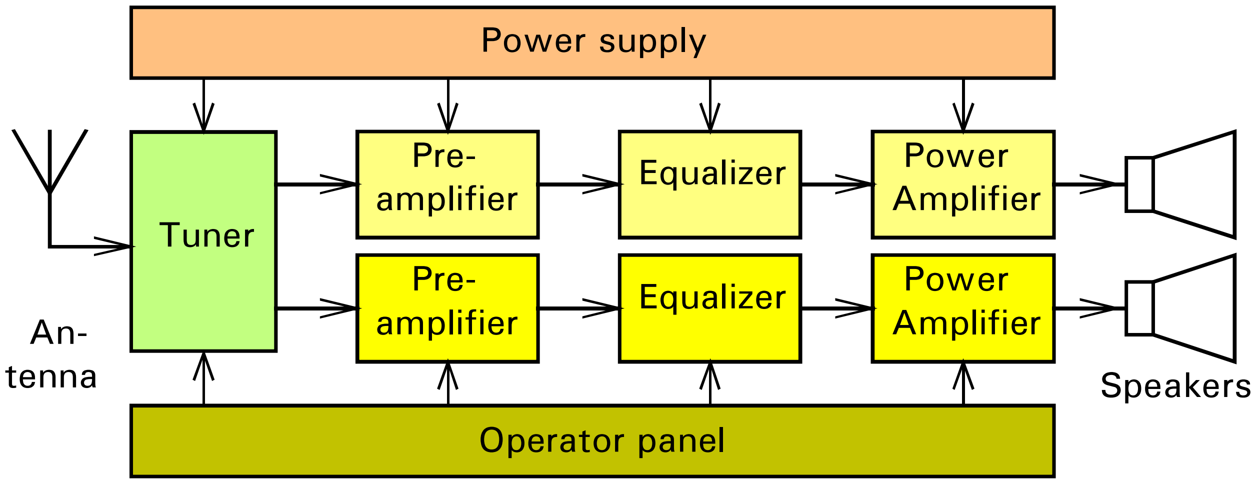
– Simultaneous engineering requires:
  * well-defined interfaces between the modules
  * well-defined information flow between design groups

4. System Architecture, Partitioning and Interfaces

Examples

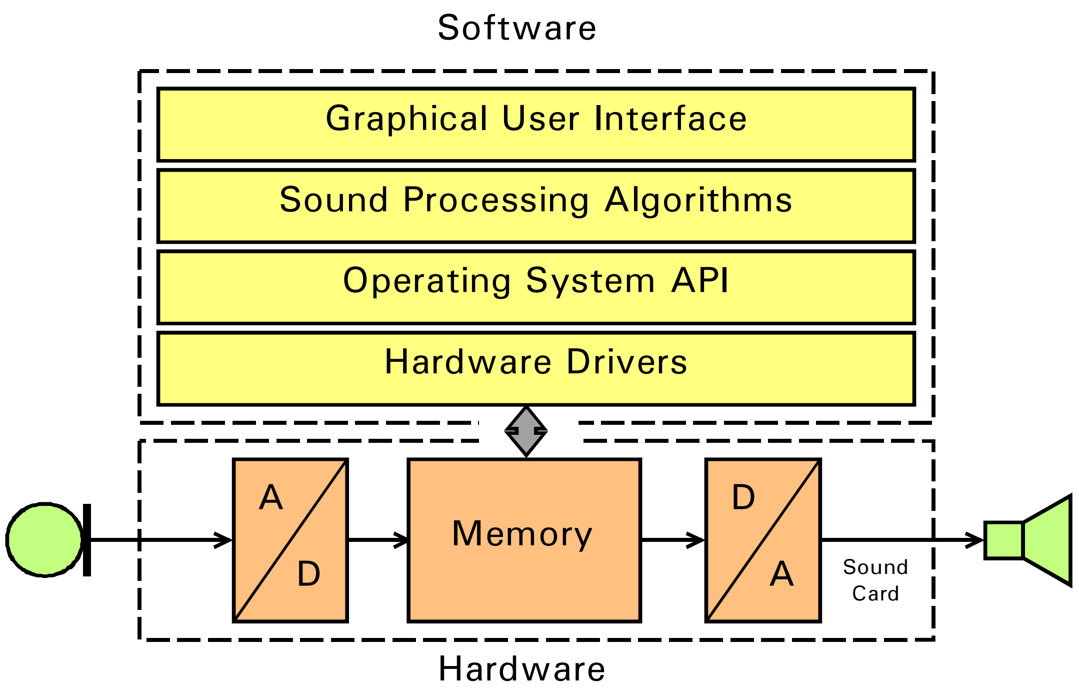
Horizontally and vertically structuring a system in blocks (modules) and layers.

- Stereo radio set



Horizontally structured (block diagram structure)

- PC based sound processing



Vertically structured ("layered")

Partitioning / structuring principles

- Signal flow centered partitioning
  - combine functions which lie in the same signal path (signal flow centered) e.g. left channel – right channel
- Function oriented partitioning:
  - Combine similar functions

    e.g. preamplifier – equalizer - ...

    e.g. physical layer – core logic – user interface

Design rules for partitioning and interfaces:

- Modules shall have clearly defined functions and interfaces
- Modules shall be independent (during development, manufacturing, test, maintenance, ...)
- Separate "stable" and "variable" functions
- Realize in software, what needs to be field upgradable
- Interfaces shall be "small and simple" (low number of signals, parameters, ...)
- Interfaces shall be robust (hardware: short circuit / overvoltage protected, software: parameter checking, error codes, ...)
- Don't reinvent the wheel, use established standards for interfaces and standard hardware/software modules (where applicable, especially in low volume products)!

Design and partitioning rules apply equally to hardware and software → see Software engineering module in Information Technology.