Requirements

Concepts

- Requirement:
  - Abstract statement of a service or constraint
  - Formal definition of a system function

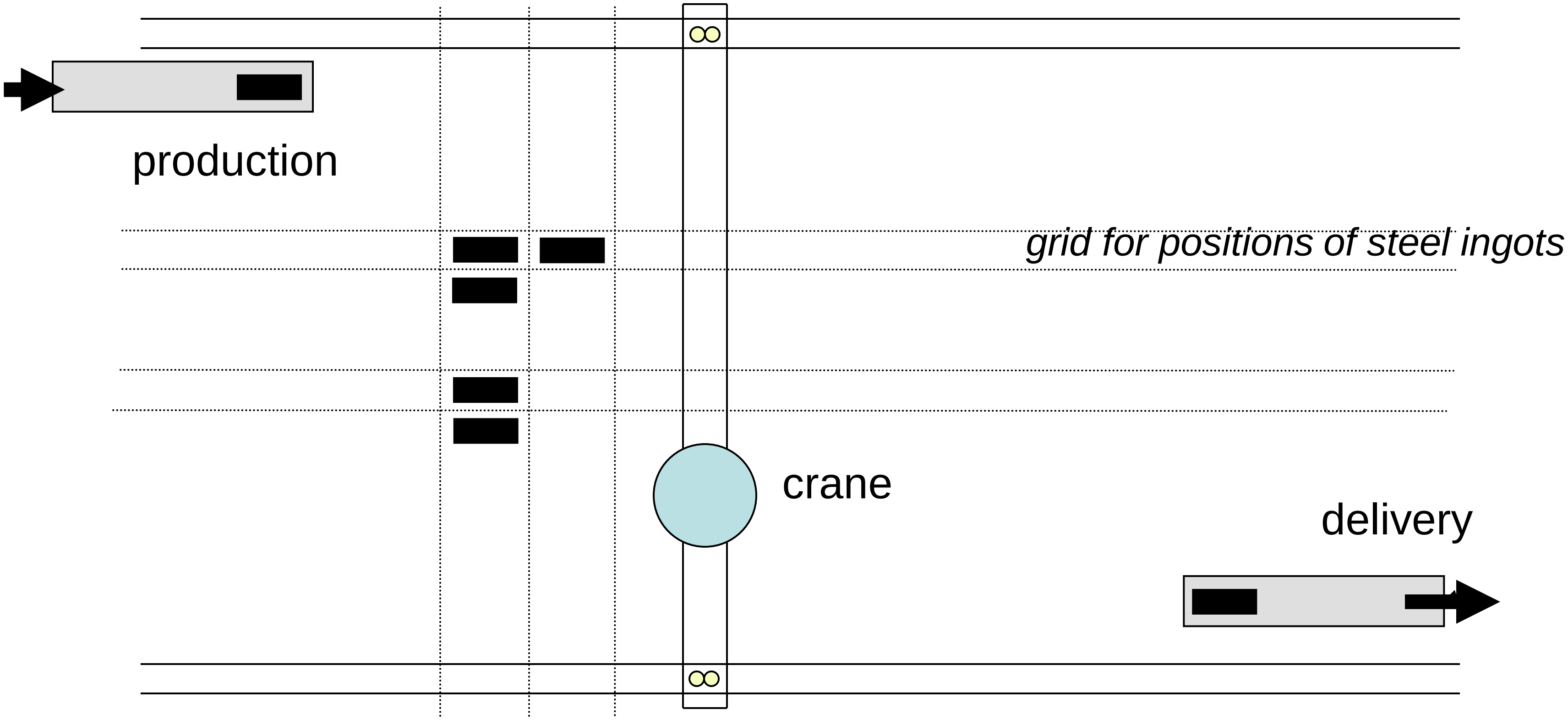
- Expressed in *natural language* + *diagrams* to *detailed mathematical functional specifications*

- Types:
  - User requirements and system requirements

    (allso called system specifications)
  - Functional, non-functional, and domain requirements;

Case study: ADMSys

- Software that controls the cooling area (deposit) of a steel factory



Example: ADMSys

Advanced Deposit Management System

- User requirement definition

1. ADMSys shall consistently operate the crane of the deposit.

- System requirement definition

1.1 ADMSys shall operate in manual and automatic modes.

1.1.1. In automatic mode, ADMSys shall issue orders for the crane and shall be notified when changes in the crane status occur. Orders are for depositing an ingot of metal and for delivering it.

1.1.1.1. For depositing a ingot, ADMSys shall decide on the position of that ingot on the deposit floor.

1.1.1.2. ADMSys shall issue an order “Go and get ingot” to the crane.

1.1.1.3 ADMSys shall wait for the crane to notify that it has loaded the ingot.

. . .

Classification of requirements

- Functional requirements
  - Services, Reaction to inputs, Behaviour in particular situations
- Non-functional requirements
  - Constraints: timing, standards, constraints on development
- Domain requirements
  - Characteristics of the application domain
  - Functional and non-functional

Functional requirements

- describe what the system should do
- should be:
  - complete (contain all services requested)
  - consistent (no contradictory definitions)
- are:
  - ambiguous
  - imprecise

Example: ADMSys

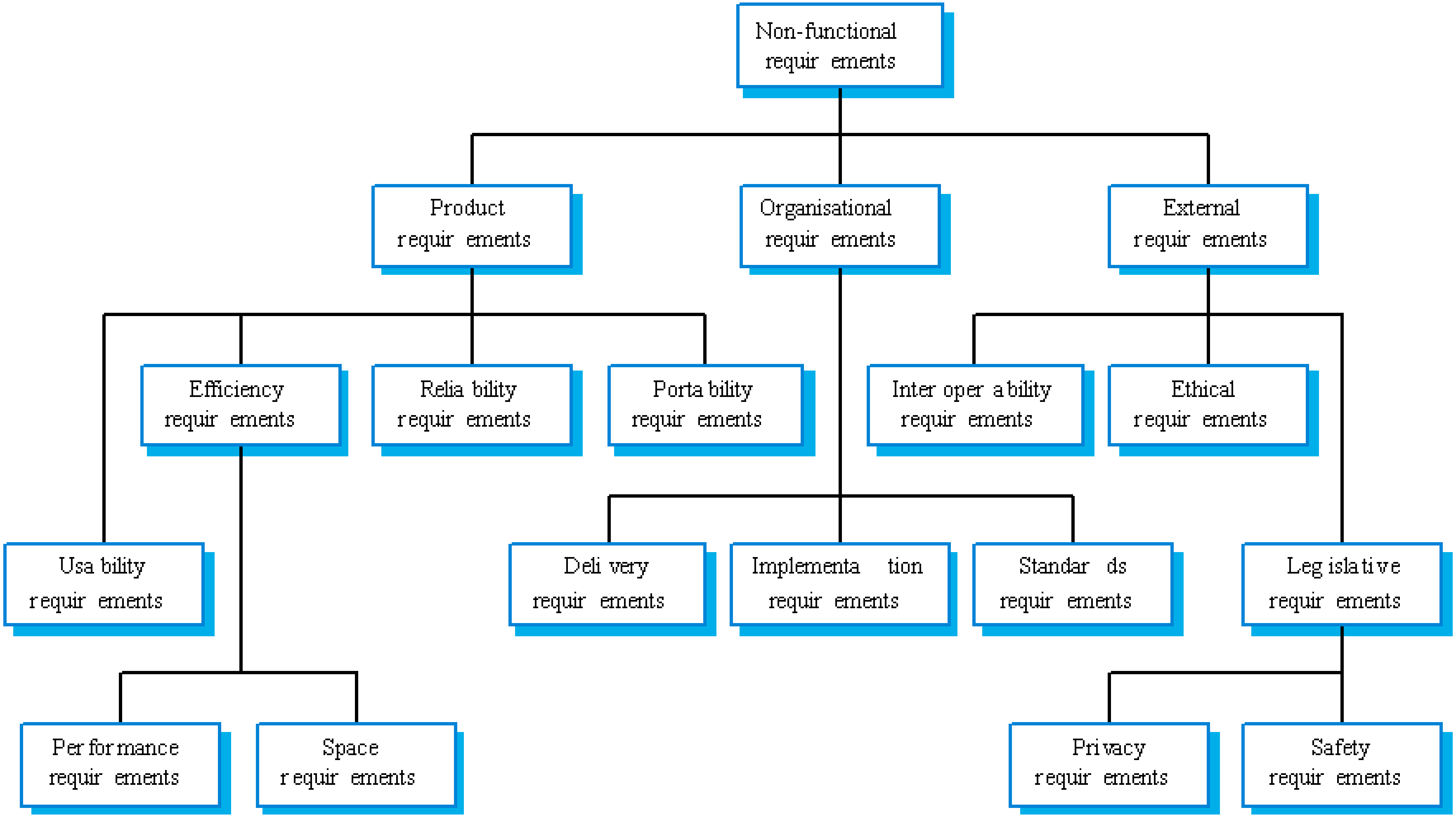
1. ADMSys shall store information about all blooms in a database.
2. The system shall update this information according to the changes in the status of the ingots.
3. The system shall execute orders from operators, in manual mode.
4. The system shall send orders to, and wait for events from, the crane controller.

1. needs further specifications (what information, database technology, etc.).
2. list with possible status values needed.
3. what are the orders? what does “execute” mean?
4. communication interface with the crane controller needs to be defined.

Non-functional requirements

- Relate to system properties (performance, reliability, availability, safety, response time, use of storage space)
- Define constraints on the software (e.g., data representation)
- *Define constraints on the software process*
- Sometimes critical

Non-functional requirements



Example: ADMSys

Product requirements

7.3. The user interface for ADMSys shall be suited for screens with resolution 1024x768 or larger.

Organisational requirements

12.1.1. The system shall be delivered as a package compatible with Microsoft Installer.

External requirements

5.4. The system shall allow operators to change a limited range of attributes of pieces, as given in Appendix 5.

Non-functional requirements

- Goal:
    - – a general intention of the user, e.g. ease of use.
    - – convey the intentions of the system users

→attributes (usability, efficiency, resilience, etc.)

- Should be verifiable
    - – define some measure that can be objectively tested.

Example: ADMSys

System goal

ADMSys should be easy to use by an experienced operator and should be organised such that the user errors are minimised.

Verifiable non-functional requirement

Experienced operators should be able to use ADMSys after two days of training. The average of errors, after this training, should not exceed two per operator per shift.

Non-functional requirements: metrics

Property	**Measure**
Speed	Processed transactions/second User/Event response time Screen refresh time
Size	M Bytes Number of ROM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems

Domain requirements

- Come from the application domain
- Very specialized, very specific terminology
- Hard to grasp
- Difficult to relate with other system requirements

The communication with the crane will use TWINCat, the Beckhoff standard protocol for accessing and controlling Beckhoff PLC.

User requirements

- provide a description of the problem from the user’s point of view.
- describe functional and non-functional requirements in such a way that they are *understandable by users without detailed technical knowledge*.
- defined using natural language, tables and diagrams.

User requirements: problems

- Lack of clarity
  - Precision is difficult without making the document hard to read.
- Confusion
  - Functional and non-functional requirements tend to be mixed-up.
- Amalgamation
  - Several different requirements may be expressed together.

Example: ADMSys

3.1. ADMSys shall provide an automated system for operating the deposit, as well as an operating environment with which the deposit operator shall be able to operate the crane and supervise the deposit. It shall give information about each bloom stored on the deposit floor in a “Properties” dialog.

3.2. ADMSys shall interact with the PLC of the crane by means of the TWINCat PLC server. ADMSys shall issue a limited set of commands, and shall be notified about events from a limited set.

Guidelines for expressing User Requirements

- Invent a standard format and stick to it
- Use language consistently to distinguish between mandatory and desirable requirements
- Use text editors features
- Avoid computer/software technical terms

Example: ADMSys

3.1 ADMSys features

ADMSys shall provide an automated system for operating the deposit, and an operating environment for the operator. The system shall

- decide the position in the deposit of a new ingot,
- decide the position in the deposit from where a stored ingot shall be delivered,
- ensure that the crane executes the necessary operations to carry the ingot at / from that position.

The operator shall be able to operate the crane and supervise the deposit.

3.1.1. In automatic mode, the system shall be notified that a new ingot is ready for depositing. It shall use an optimisation algorithm to choose the position of the new ingot on the deposit floor. It shall issues orders for the crane to move to the ingot, load it and carry it to the calculated position.

3.1.2. In manual mode, the operator shall control the system.

- The decision on the position of the new ingot shall remain to the operator.
- It shall be possible for the operator to order the crane to move and take the ingot.

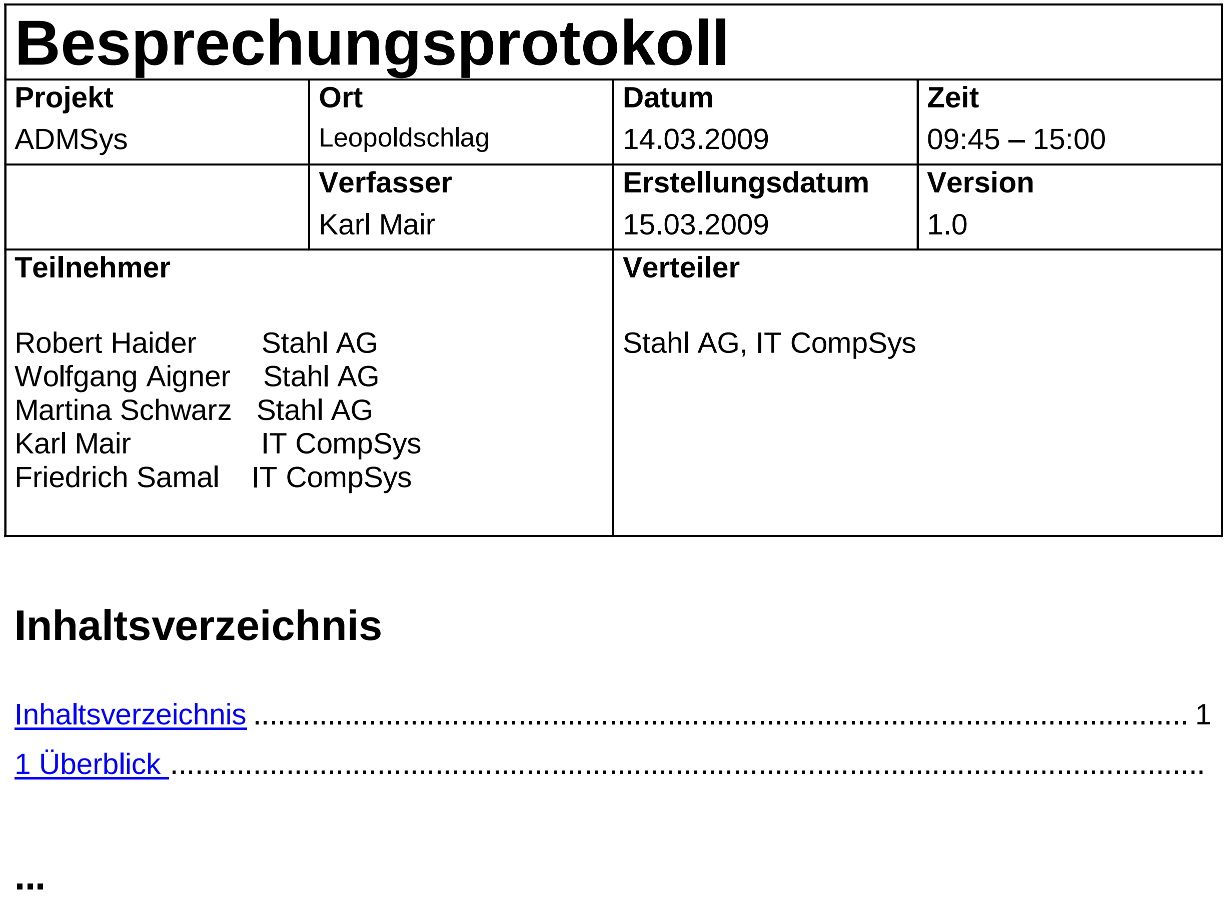
System requirements

- Expanded, structured, detailed versions of user requirements
- ***Used as basis for system design***
- Specify how the user requirements should be provided by the system
- May be incorporated into the system contract.

Gathering requirements

- Meetings with the customer
- Minutes of the meeting (Besprechungsprotokoll)
- Content:
  - Project name
  - Place of the meeting
  - Date and time of the meeting
  - Writer of the document (minutes)
  - Date of issue of the document
  - Version of the document
  - Participants to the meeting
  - Contributors (companies)
  - Topics discussed

Example: ADMSys



Besprechungsprotokoll

Projekt ADMSys	**Ort** Leopoldschlag	**Datum** 14.03.2009	**Zeit** 09:45 – 15:00
	Verfasser Karl Mair	**Erstellungsdatum** 15.03.2009	**Version** 1.0
Teilnehmer Robert Haider Stahl AG Wolfgang Aigner Stahl AG Martina Schwarz Stahl AG Karl Mair IT CompSys Friedrich Samal IT CompSys		**Verteiler** Stahl AG, IT CompSys	

Inhaltsverzeichnis

...

The Software Requirements Document

- Official statement of what is required of the system developers
- Includes
  - a definition of user requirements
  - a specification of the system requirements.
- It is NOT a design document
  - it should set WHAT the system should do rather than HOW it should do it

Intended audience

System customers	Specify the requirements and read them to check that they meet their needs. They specify changes to the requirements
Managers	Use the requirements document to plan a bid for the system and to plan the system development process
System engineers	Use the requirements to understand what system is to be developed
System test engineers	Use the requirements to develop validation tests for the system
System maintenance engineers	Use the requirements to help understand the system and the relationships between its parts

Structure (IEEE standard)

- A generic structure for a requirements document that must be instantiated for each specific system
- A general framework: to be adapted!
  - **`Introduction.`**
  - **`General description.`**
  - **`Specific requirements.`**
  - **`Appendices.`**
  - **`Index.`**

Recommended Structure

- Introduction
- Glossary
- User requirements definition
- System architecture
- System requirements specification
- System models
- System evolution
- Appendices
- Index

Introductory part

- Introduction
  - Describe the need for the system
  - Briefly describe the functions
  - Explain how it will work with other systems

Introductory part

- Glossary
  - Define technical terms used in the document

Do not make assumptions

about users' expertise!

Specifications

- User requirements definition
  - Services provided for the user
  - Non-functional system requirements
  - Product and process standards
- Use natural language, diagrams, etc.

Specifications

- System architecture (a diagram)
  - High-level overview of the anticipated system architecture
  - Show distribution of functions across modules

  - Should be understood by the client as well!

Specifications

- System requirements specifications
  - Describe functional and non-functional requirements in more detail
    - Describe user interactions, possibly as a set of scenarios.

Specifications

[

- System models
  - Contains one or more models
  - Show relationships
    - Between components
    - Between the system and its environment

- Types of models
  - Object
  - Data flow
  - …

]

Specifications

[

- System evolution
  - Describe fundamental assumptions on which the system is based
  - Anticipate changes (hardware evolution, user needs changing, etc.)

]

Final part

- Appendices
  - Specific information related to the system, in detail
    - Hardware requirements: minimal/optimal configurations
    - Database requirements: logical organisation of data

Final part

- Indices:
  - Term index (alphabetic)
  - Diagram index
  - Table index
  - etc.

C# Lecture 2

Language basics. Classes. Interfaces.

Language basics

- Variables
  - Data types (double, int, …)
  - Arrays
- Operators (*, -, &&, …)
- Expressions, statements, blocks
- Control flow statements

Language basics

- Methods
- *Properties*

```
class Polygon
{
      private double[][] points;
      public double[][] Points
      {
            get { return points; }
            set { points = value;}
      }
. . .
}
```

Syntax highlights

- The “.” operator
- The “this” keyword
- The “base” keyword

- Interface:
  - defines a *contract* that all implementing classes must comply to.
  - The contract: basically, a set of functions / properties;
    - all classes that implement the interface are expected to have those functions. They must be public.

C# classes

- Concepts:
  - Class – a means to define your own types.
  - Instances of the class - *objects*
  - Members of an object
    - Data → *states*
    - Functions → *behaviour*
  - Class members vs. instance members
  - Access modifiers (public, internal, protected, private)

Classes: an example

1. **Create a new C# project GeoTools**
   - Add a folder `BasicShapes`.
     - Shall contain geometric objects that can be e.g. drawn.
   - Add a reference to System.Drawing
   - Most geometric objects should:
     - Contain some points,
     - Be able to respond to some queries (parameter, area, etc.),
     - Be able to export a `string` representation,
     - Be able to draw themselves.

Interfaces

- → All geometry objects should expose a common interface
  - – Add a new item to folder BasicShapes – an interface file named IGeoObject.

```
using System;
namespace GeoTools.BasicShapes
{
    using System.Drawing;
    public interface IGeoObject
    {
       double[][] Points { get; }
       double Perimeter { get; }
       double Area { get; }
       void Draw(Graphics g);
    }
}
```

Classes implement interfaces

- Add a new item to folder BasicShapes – an Class file named Square.

```
using System;
namespace GeoTools.BasicShapes
{
   using System.Drawing;
   public Square: IGeoTools {
        . . .
        public Square(double[] upperLeftCorner,
                double width)
        {
        . . .
        }
        public double[][] Points
        {
                get { . . . }
        }
    . . .
}
```

Homework

- Identify problems (undefined/unclear concepts, questions, documents needed, different levels of abstraction, etc.) in the following user requirements:

> All finalized orders are collected in the warehouse and the delivery companies in charge with the transport of these orders are notified. The following information shall be displayed on the screen for the definition of loading orders: order number, client number, client name, PO code of the delivery address, delivery date, number of pallets, total weight, whether the client order is complete. When the truck driver arrives at the reception, the corresponding orders are selected and a delivery document with all of them shall be generated, with a specific delivery number.