Motivation and Contents Overview

Software Engineering I
Fall semester 2006/2007

Department of Computer Sciences
cs.uni-salzburg.at

Dr. Stefan Resmerita

Outline of the material for the courses Software Engineering I and II

UNIVERSITÄT SALZBURG

Goals

- Learning the mostly used approaches to software development (in the small and in the large)
- Developing an understanding of what is good and what is bad software (-construction)
- Knowing and understanding concepts and terms
- Developing a first understanding of the „Software development in the large“

Example isn‘t another way to teach, it is the only way to teach

Albert Einstein

SE I (1)

- **Concepts and constructs for flexible software**
  - Frameworks and Design Patterns
  - Software parameterization (configuration files, resources, script languages)
  - Heuristics for adequate flexibility
  - Model-driven architecture (MDA) of OMG

UNIVERSITÄT SALZBURG

SE I (2)

- **Concepts and constructs in Component-Based Design**
  - **The Module concept**
  - **Overview of standards for components (WebServices, JavaBeans, OSGi)**
  - **Heuristics for adequate modularization (Balance between Coupling and Cohesion in a Discrete Event Simulation example)**
  - **Methods for analysis of software architectures**
  - **Aspect Oriented Programming(AOP)**

UNIVERSITÄT SALZBURG

SE II (1)

- Transformational software development
  - Concepts for design systematization and for automatization of the implementation:
    - Formal Languages
    - Attribute Grammars

UNIVERSITÄT SALZBURG

SE II (2)

- Process model
- Software quality management
- Legacy systems, re-engineering
- Software metrics
- Testing and verification
- Software development
- Modelling methods and tools
- Configuration management

Software Technology: State of the Art and Challenges

Software Engineering I
Fall semester 2006/2007

Department of Computer Sciences
cs.uni-salzburg.at

Dr. Stefan Resmerita

UNIVERSITÄT SALZBURG

Context

- The phenomenon Software
- How can Software be engineered?
- Software techniques – Quo vadis?

The Phenomenon Software

UNIVERSITÄT SALZBURG

The Computer as universal machine makes Software pervasive

Airplane/Rocket control

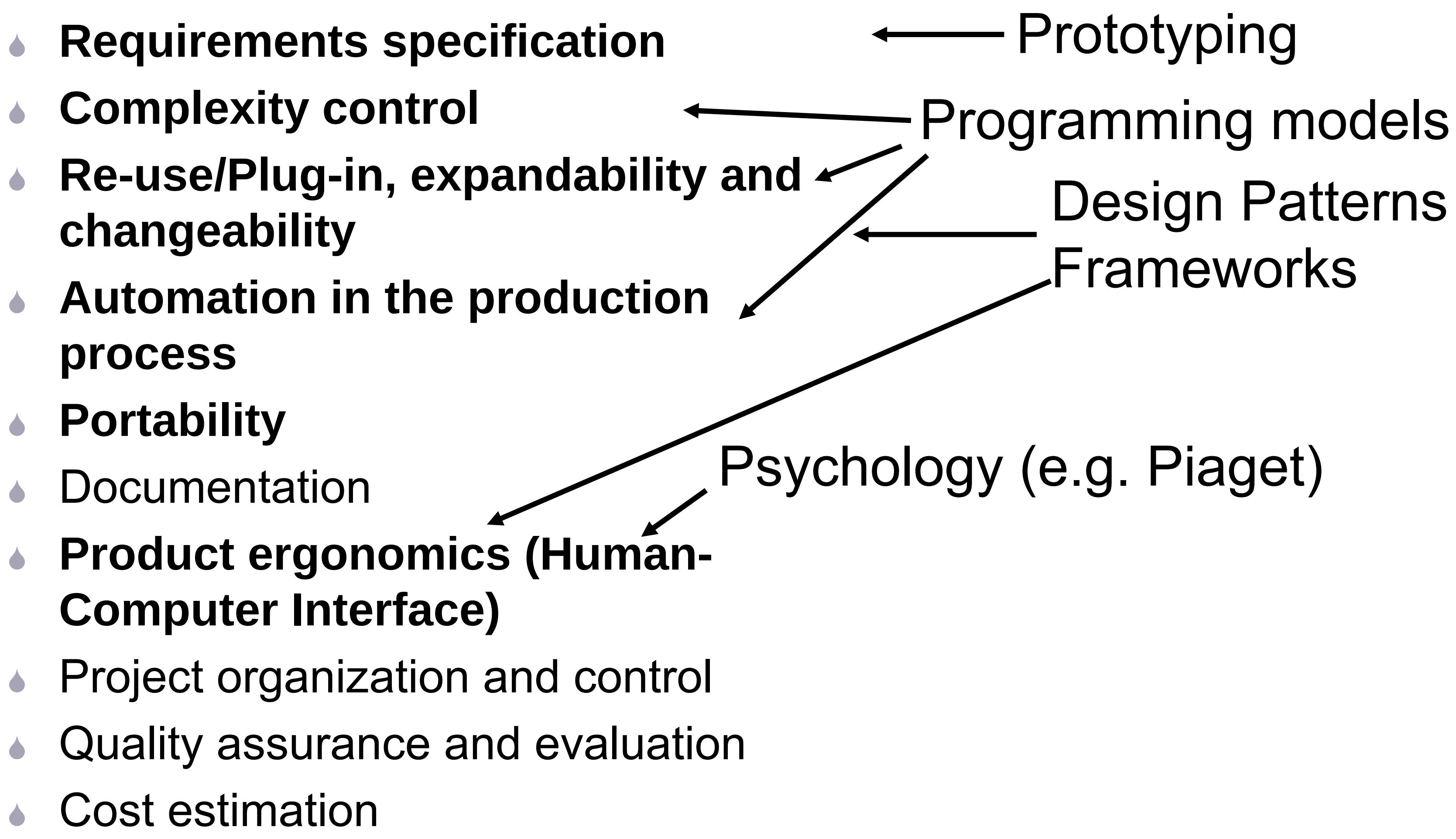
ca. 70 Processors
in a car

UNIVERSITÄT SALZBURG

What is so special about Software?

UNIVERSITÄT SALZBURG

The problems with software production is the complexity of the achieved product



- **Requirements specification**
- **Complexity control**
- **Re-use/Plug-in, expandability and changeability**
- **Automation in the production process**
- **Portability**
- Documentation
- **Product ergonomics (Human-Computer Interface)**
- Project organization and control
- Quality assurance and evaluation
- Cost estimation

Prototyping

Programming models

Design Patterns

Frameworks

Psychology (e.g. Piaget)

UNIVERSITÄT
SALZBURG

Quality problems

- Software bugs: deficiencies with drastic effects
  - Incorrect bank transactions
  - Y2K
  - Ariane
  - Mars adventures
    - PathFinder
    - Spirit

UNIVERSITÄT SALZBURG

Human interaction problems

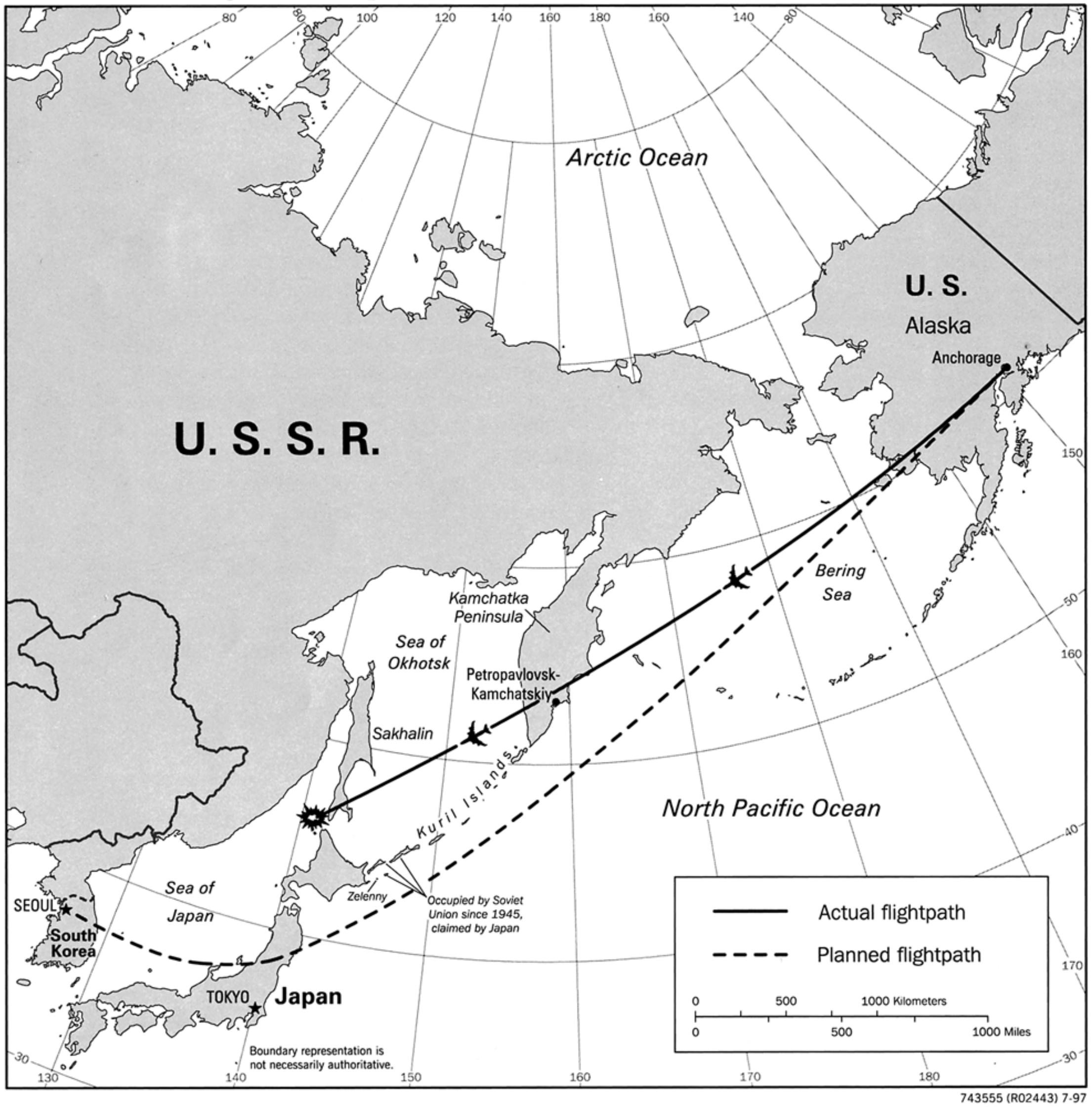
- Human-Computer Interaction

- Human-Machine Interaction
  - Interaction with automated systems
  - Example: Korean Air Lines Flight 007

- Computer pervasiveness makes the human interaction issue very important

KAL007

Korean Airlines Flight 007, 1 September 1983



Example: Specification problems

UNIVERSITÄT SALZBURG

An exact specification is often impracticable

given.: $n \geq 3$,

$$L\colon N_n \to N$$

find.: A Program P that computes

$$a\colon N_3 \xrightarrow{\text{inj}} N_n \text{ , such that}$$

$$\bigwedge_{1 \leq i \leq 3} \; \bigwedge_{\substack{j \in N_n \setminus \cup \{ a_k \} \\ 1 \leq k \leq j}} L(a_i) \geq L(a_J)$$

...while a verbal specification is often inaccurate

Given a list with at least three positive numbers

Find a program P that gives the indices of the three largest elements of the list.

Mastering Complexity

UNIVERSITÄT SALZBURG

In classical engineering disciplines

- Bad quality can hardly be hidden
  - Door cannot close well
  - Unnecessary artifacts
    - „Fifth wheel to the car“
- Resources are limited
  - Engineering approaches mean optimization under the given basic conditions

Bad quality is not so visible in software

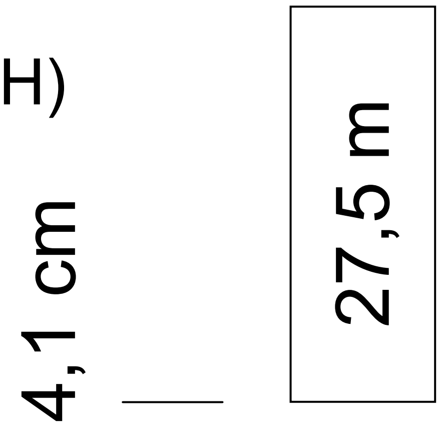
- Bad structuring
  - „Spaghetti“ program code:
    - Wheel change -> the motor works no more
  - Replicated program code
- Hardly re-usable code
  - The wheel is always re-invented

UNIVERSITÄT SALZBURG

Engineering procedures do not seem to pay off

- Hardware resources evolve according to Moore‘s Law; thoughtless handling of this issue leads to:
  - Unnecessary complexity
  - No longer understandable artifacts

OberonOS (ETH ZH) 30.000 lines of program code



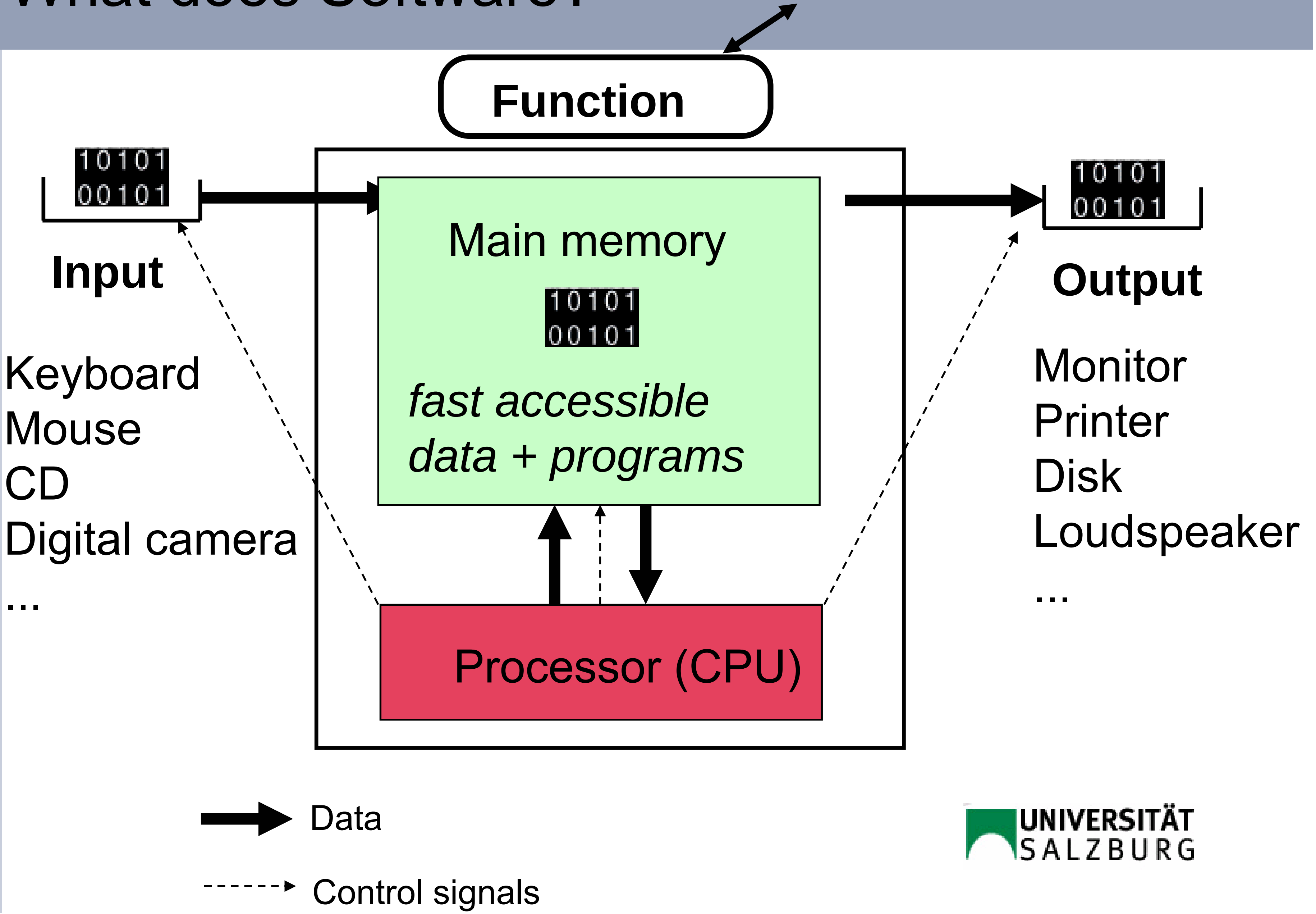
4,1 cm

27,5 m

Windows XP (2002): 40.000.000 (!!) lines of program code

How can Software be engineered?

What does Software?



UNIVERSITÄT
SALZBURG

Interaction with the environment

- Interactive systems: the computer is the leader of the interaction
  - Examples: Operating systems, Database systems
  - Main issues: Deadlock, Fairness

- Reactive systems: the environment is the leader of the interaction
  - Examples: Industrial process control, airplane control
  - Main issues: Safety, Timeliness

Processing of photos from digital camera

90 °

Colors off

More examples

- ABS in automotive
  - **Input:** Rotational speeds of the wheels and user braking
  - **Function:** Checking whether the speeds are zero when the user brakes
  - **Output:** Appropriate controlling of the braking force
- Bank transfers
  - **Input:** Transfer data (payee, payer, amount)
  - **Function:** Validation of the transaction
  - **Output:** New transaction lines in the accounts