Master of Science in Computer Science

University of Paderborn
Gerd Szwillus, Head of the Examination Committee
April 2015
Outline

- Prerequisites
- Elements of the Master Programme
- Structure
- Scope
- Examinations
Prerequisites of the Masters Program

- What you learnt in your Bachelor’s degree …
  - in India, Syria, Egypt, Pakistan, …

- … must match (roughly) what Paderborn students learn in their Bachelor’s degree
  - PB Bachelor provides the necessary requisites

- Most important check done before admission!
  - We want that you succeed!
Prerequisites of the Masters Program

- Paderborn Bachelor
  - mostly done in German
  - content organization analogous to Master
  - same teaching personnel
  - teaching goals match prerequisites

- Bachelor and Master
designed as a consecutive study program
Prerequisites of the Masters Program

- Ability to apply foundations of Computer Science
- Experience in Doing Practical Work
- Ability to do Scientific Work
Prerequisites: Ability to apply foundations of CS

- usage of **formal calculi** to specify problems, structures, systems, languages, ...
- formal methods to **analyze algorithms**
- formal methods to **check solutions**
Prerequisites: Experience in Doing Practical Work

- application of methods and tools for software design and implementation
- substantial experience in at least one programming language
- ability to switch to a new programming language within short time
Prerequisites: Ability to do Scientific Work

- searching for and investigating scientific publications

- writing scientific documents
  - adequate structure
  - clear descriptions and explanations
  - citations and references
  - correct English

- developing and performing presentations
Elements of the Masters Program

- Seminar (Seminar)
  - 2 meeting hours per week (S2)
  - Teacher proposes topics (Seminarthemen)
  - Every student
    - selects one topic to work on
    - prepares a talk with slides (Seminarvortrag)
    - submits a written elaboration (Ausarbeitung)
Elements of the Masters Program

- **Project Group** *(Projektgruppe)*
  - **Workload**: Half-time for one year
  - PG topic presentation in a public event *(Projektgruppenvorstellung)* in the last week of teaching season
  - Interested students apply for a PG and 8 to 16 students are accepted
  - Working for two semesters on a project (often: concept and implementation of some software)
  - Highly self-organized
  - **All** our project–groups are in English!
Elements of the Masters Program

- Master’s Thesis (*Masterarbeit*)
  - **Workload**: Full-time for one semester
  - What has to be done?
    - Literature review
    - Research
    - Often: implementation of some software
    - Writing a thesis (~ 80–120 pages) on scientific level
    - All this: within 6 months (formally checked!)
  - First a **planning** phase (typically 1 month)
  - Then the **performing** phase (5 months)
Elements of the Masters Program

- Master’s Thesis *(Masterarbeit)*
  - Finding an advisor *(Betreuer)*
    - adress professors working in an interesting area
    - adress project group organizer
    - ask friends
  - Finding a topic *(Thema)* is an interactive process between student and advisor
    - own ideas?
    - project group topic?
    - discussion with the (potential) advisor!
German Language Course (*Deutschkurs*)

- Hosted by the International Office *(Akademisches Auslandsamt)*
- Obligatory *(no Master‘s degree without!)*
- Typically covers the „General Studies“ part
- Check
  - http://www.upb.de
  - International Students
  - EN (for English)
  - International Office
  - German Language Courses
  - German for international degree students
Four areas (Gebiete)

- **SWT&IS**: Software Technology & Information Systems (Softwaretechnologie und Informationssysteme)
- **MuA**: Models & Algorithms (Modelle und Algorithmen)
- **ESS**: Embedded Systems & System Software (Eingebettete Systeme und Systemsoftware)
- **MMWW**: Human–Machine–Interaction (Mensch–Maschine–Wechselwirkung)

Every student chooses one of the four as area of specialization (Vertiefungsgebiet)
Structure

- Modules (*Module*)
  - Every module
    - belongs to one area (of the four)
    - contains a catalog of classes or seminars (from that area)
    - is described in the module handbook (*Modulhandbuch*)
    - is studied by taking *two* classes from that catalog
  - Every student has to
    - study *three* modules in her/his *area of specialization*
    - *one* module each in the *three other areas*
    - (Hence every student has to take *six* modules overall)
Structure

- Handbook of modules (Modulhandbuch, MHB)
  - available on-line as PDF from http://www.cs.uni-paderborn.de

  ➔ English

  ➔ Students

  ➔ Examinations

  ➔ Module Handbook

III. Modules in the Masters Degree Course

III.1 Field: Software Technology and Information Systems

III.1.1 Software Technology I

Role in the computer science degree course
Software technology deals with concepts, languages, methods, and tools for developing and maintaining large software systems. In this context, we pay substantial attention to the quality of the software systems. In particular, this attention involves ensuring that computer scientists meet the functional and non-functional requirements for the software systems. Depending on the area of application, we place a different emphasis on the individual system requirements. Examples include safety requirements in embedded systems or usability requirements in interactive systems.

The classes in this module deal in particular with techniques for developing embedded systems and software for applications in the engineering disciplines (from a University of Paderborn perspective, these are in particular mechanical and electrical engineering). The emphasis is on software development considered in terms of sufficiently high safety and correctness requirements and of the resource limits that are typically present in such systems. The module gives the students a deeper understanding of how such specific software systems may be built or extended. This goal requires that the student acquire specific modeling and implementation concepts and languages, which are practised on larger application examples.

The classes of the module build in particular on the contents of the first stage modules in Software Technology and Information Systems (I.1) and on the class Model-Based Software Development (MBSD) of the Software Technology and Information Systems.
### Structure

- Currently existing modules

<table>
<thead>
<tr>
<th>Modules in the Master Degree Course</th>
</tr>
</thead>
<tbody>
<tr>
<td>III.1 Field: Software Technology and Information Systems</td>
</tr>
<tr>
<td>III.1.1 Model-based software development (M.079.3111)</td>
</tr>
<tr>
<td>III.1.2 Languages and Programming Methods (M.079.3112)</td>
</tr>
<tr>
<td>III.1.3 Databases and Information Systems (M.079.3113)</td>
</tr>
<tr>
<td>III.1.4 Knowledge-Based Systems (M.079.3114)</td>
</tr>
<tr>
<td>III.1.5 Analytical Methods in Software Engineering (M.079.3115)</td>
</tr>
<tr>
<td>III.1.6 Constructive Methods in Software Engineering (M.079.3116)</td>
</tr>
<tr>
<td>III.2 Field: Models and Algorithms</td>
</tr>
<tr>
<td>III.2.1 Algorithms I (M.079.3211)</td>
</tr>
<tr>
<td>III.2.2 Algorithms II (M.079.3212)</td>
</tr>
<tr>
<td>III.2.3 Complexity and Cryptography (M.079.3213)</td>
</tr>
<tr>
<td>III.2.4 Algorithms in Computer Networks (M.079.3214)</td>
</tr>
<tr>
<td>III.3 Field: Embedded Systems and System Software</td>
</tr>
<tr>
<td>III.3.1 Distributed Computer Systems (M.079.3311)</td>
</tr>
<tr>
<td>III.3.2 System Software (M.079.3312)</td>
</tr>
<tr>
<td>III.3.3 Computer Networks (M.079.3313)</td>
</tr>
<tr>
<td>III.3.4 Embedded Systems (M.079.3314)</td>
</tr>
<tr>
<td>III.3.5 HW/SW Codesign (M.079.3315)</td>
</tr>
<tr>
<td>III.3.6 Embedded and Real-Time Systems (M.079.3316)</td>
</tr>
<tr>
<td>III.4 Field: Human-Machine Interaction</td>
</tr>
<tr>
<td>III.4.1 Computer Graphics and Visual Computing (M.079.3421)</td>
</tr>
<tr>
<td>III.4.2 Computer Science and Society (M.079.3422)</td>
</tr>
<tr>
<td>III.4.3 Accessible Human-Computer Interaction (M.079.3423)</td>
</tr>
<tr>
<td>III.4.4 Computer-Supported Cooperative Work and Learning (M.079.3424)</td>
</tr>
<tr>
<td>III.4.5 User Interface Development (M.079.3425)</td>
</tr>
<tr>
<td>III.4.6 Model-Based Development of User Interfaces (M.079.3426)</td>
</tr>
</tbody>
</table>
Structure

- Example: Module III.2.1 „Algorithms 1“

- Studying this module means: taking two of these classes!

- This is a general principle: Select two classes from a catalogue

Content structure of the module
The following classes are offered:

- Approximation Algorithms
- Randomised Algorithms
- Online Algorithms
- Algorithmic Game Theory
- Algorithmic Coding Theory I
- Network Flow Algorithms
- Graph Algorithms
- Geometric Algorithms
Structure

- Check „Schedule of courses“
- Classes selectable for Module III.2.1
- Table available from the same page as the module handb.
Scope

Every student has to ...

- study **three** modules in her/his area of specialization (24)
  - including **one seminar** in some module
- study **one** module in each of the other three areas (24)
  - one more seminar (instead of a class) is allowed
- successfully take part in a project group (30)
- write a **Master‘s thesis** (30)
- complete the German **language** course (12)
You have the choice!

- Modules can be scheduled within one semester...
- ... or their two classes can be spread over more semesters

- Example student with ESS as specialization area might take his 2 classes in MuA, SWT, MMWW and his 6 classes in ESS like this →
Schedule – Example

- MuA₁ & MuA₂ (from one module) in sem. 1 and 3
- SWT₁ & SWT₂ in sem. 1
- MMW₁ & MMW₂ in sem. 1 and 3
- ESS₁…ESS₆ (matched to three modules), e.g. ESS₁ & ESS₄  ESS₂ & ESS₃  ESS₅ & ESS₆
Good idea (not always possible):
Do a lot of classes in the **specialization area** first…
  ◦ … then a project group
  ◦ … then a thesis in this field
→ „grow“ into your favourite subject

Consider workload!
  ◦ 30 ECTS points is the officially recommended average workload
  ◦ Our example student‘s workload:
    20 (+12 German) + 23 + 31 + 34 = 120
Examinations

- Modules
  - one oral exam about the two classes (resp. a class and a seminar) – no exams about **single classes**!
  - module exam only after **both classes** have been visited!

- Project group
  - permanent evaluation throughout the project

- Master’s Thesis
  - evaluation from the advisor and a co-advisor

- Language Course
  - Language certificate after the course
Examinations

- Repetition
  - Oral exams (modules) can be repeated **twice**
  - Master‘s Thesis can be repeated **once**
  - Project group can be repeated **twice** (you will not want that!)
  - German Language Course can be repeated **infinitely**
Examinations

- Compensation

There are (limited) ways

- to shift aside („compensate“) exam **failures**
  - e.g. failure in one module
  - give up on that one and try another one instead

- to **improve** exam results
  - e.g. you succeeded in a module exam but with an unsatisfactory result, you can
  - try a different module additionally
Examinations

- Final failure is possible!
  - you have no more option to use compensation and you are no more able to fulfill the requirements
    - 3 modules in the specialization area
    - one module in each of the other areas
  OR
  - three attempts for a project group failed
  OR
  - two attempts for a Master‘s thesis failed

😊
Examinations

- Final failure is possible!
  - you have no more option to use compensation and you are no more able to fulfill the requirements
    - 3 modules in the specialization area
    - one module in each of the other areas
  - OR
    - three attempts for a project group failed
  - OR
    - two attempts for a Master's thesis failed

But this will NEVER happen, of course!!!!!
Welcome in Paderborn!

- We wish you a successful and enriching study experience!
- Questions?
  - Now!
  - Anytime: study-advice.cs@upb.de