Welcome!

CS1000
Explorations in Computing
Department of Computer Science
Michigan Technological University

Dr. Nilufer Onder
Fall 2015
Fisher 139
Outline

- Information about me
- Tips to connect with faculty
- Course information
- Computer Science curricula
Short biography

- BSc in Computer Engineering
  Orta Dogu Teknik Universitesi
- MSc in Computer Engineering
  Orta Dogu Teknik Universitesi
- Worked as a systems analyst
- PhD in Computer Science
  University of Pittsburgh
- Came to Michigan Tech in 1999
Who motivated me

- My parents and family
- Faculty, advisors, bosses
Research Area

Computer Science

Artificial Intelligence

Planning

Decision Making Under Uncertainty
Courses I teach

- CS1000 – Explorations in Computing undergrad, required
- CS 3311 - Formal Models of Computation undergrad, required
- CS 4811 – Artificial Intelligence undergrad, elective
- CS 5811 – Advanced Artificial Intelligence grad
- SSE 3200, CS 3090 – Web Based Services undergrad
Outline

- Information about me (done)
- Tips to connect with faculty
- Course information
- Computer Science curricula
Tips to connect with faculty

- Don't hesitate to initiate conversations with your professors
- Lots of professional advantages to getting to know each other
- Logistics, scheduling
Outline

- Information about me (done)
- Tips to connect with faculty (done)
- Course information
- Computer Science curricula
CS1000

- “Explorations” in Computing
- Explorations that lead to success in
  - Academic life
  - Career planning
- Forward looking course
- Check out the course syllabus
Outline

- Information about me  (done)
- Tips to connect with faculty  (done)
- Course information  (done)
- Computer Science curricula
Software engineering degree flowchart

**Software Engineering (SSEN)**

**Degree Flowchart**

**127 Credits**

**Effective Fall 2015**

### Year 1
- **Fall**
  - CS1000 (1) Explorations in Computing
  - MA1031/1032 Calculus I
  - UN1025 Global Issues or 3K-level Language
- **Spring**
  - MA1160 Intro Prog. I
  - MA2160 Calculus II
  - Technical Elective (3) [f]

### Year 2
- **Fall**
  - MA1160/1161 Discrete Structures
  - CS2311 (3) Intro Prog. II
  - CS1122/1131 Intro to Algorithms
- **Spring**
  - CS2321 (3) Data Structures
  - MA2720 (4)/MA3710 (3) Statistics
  - CS3425 (3) Databases

### Year 3
- **Fall**
  - CS1122/1131 C for JAVA prog.
  - MA1160/1161 Intro Prog. III
  - MA2330 (3) Linear Algebra
- **Spring**
  - CS3421 (4) Comp Org
  - CS3141 (3) Team Software Proj
  - CS4710 (3) Model-driven SW Dev.

### Year 4
- **Fall**
  - CS3121/3131
  - CS2311/2321
  - CS3311 Formal Models of Comp
  - CS3331 (3) Concurrent Computing
- **Spring**
  - CS4712 (3) SW Qual. Assurance
  - CS4760 (3) Human-Comp. Interface
  - CS4711 (3) SW Proc and Mgmt
  - Technical Elective (3) [f]
Question

- Where do curricula come from?
Computer Science Curricula 2013

Curriculum Guidelines for Undergraduate Degree Programs in Computer Science

December 20, 2013

The Joint Task Force on Computing Curricula
Association for Computing Machinery (ACM)
IEEE Computer Society

A Cooperative Project of
What to consider when designing a curriculum?

- Reflects the state of the art body of disciplinary knowledge (reasonable size)
- Is rigorous
- Is flexible to meet needs of individual departments and students
- Is pedagogically sound and complete
- Has good breadth and depth coverage
- Considers input and feedback from a broad community
- Revised continually
Characteristics of CS graduates

- Technical understanding of computer science
- Familiarity with common themes and principles
- Appreciation of the interplay between theory and practice
- System-level perspective
- Awareness of the broad applicability of computing
Characteristics of CS graduates (cont'd)

- Problem solving skills
- Project experience
- Commitment to life-long learning
- Commitment to professional responsibility
- Communication and organizational skills
- Appreciation of specific knowledge in other domains (cross disciplinary)
Knowledge areas and core hours

- Technologies change rapidly over time
- Essential concepts, perspectives, and methodologies that are constant define computer science
- The body of knowledge is organized into 18 knowledge areas. For each
  - Tier 1: essential for all CS programs
  - Tier 2: individual programs choose their coverage
- Knowledge areas are not intended to describe specific courses
<table>
<thead>
<tr>
<th>Knowledge Area</th>
<th>Tier 1</th>
<th>Tier 2</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithms and complexity</td>
<td>19</td>
<td>9</td>
<td>28</td>
</tr>
<tr>
<td>Architecture and Organization</td>
<td>0</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>Computational Science</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Discrete Structures</td>
<td>37</td>
<td>4</td>
<td>41</td>
</tr>
<tr>
<td>Graphics and Visualization</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Human-computer interaction</td>
<td>4</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>Information assurance and security</td>
<td>3</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>Information management</td>
<td>1</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>Intelligent systems</td>
<td>0</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Networking and communication</td>
<td>3</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>Operating systems</td>
<td>4</td>
<td>11</td>
<td>15</td>
</tr>
<tr>
<td>Platform-based development</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Parallel and distributed computing</td>
<td>5</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>Programming languages</td>
<td>8</td>
<td>20</td>
<td>28</td>
</tr>
<tr>
<td>Software development fundamentals</td>
<td>43</td>
<td>0</td>
<td>43</td>
</tr>
<tr>
<td>Software engineering</td>
<td>6</td>
<td>22</td>
<td>28</td>
</tr>
<tr>
<td>Systems fundamentals</td>
<td>18</td>
<td>9</td>
<td>27</td>
</tr>
<tr>
<td>Social issues and professional practice</td>
<td>11</td>
<td>5</td>
<td>16</td>
</tr>
<tr>
<td><strong>Total core hours</strong></td>
<td><strong>165</strong></td>
<td><strong>143</strong></td>
<td><strong>308</strong></td>
</tr>
</tbody>
</table>
Totals

- All Tier1 + All Tier2  308 (8 courses)
- All Tier1 + 90% of Tier2  294 (7 courses)
- All Tier1 + 80% of Tier2  280 (7 courses)
Software engineering degree flowchart

Software Engineering (SSEN) Degree Flowchart 127 Credits Effective Fall 2015

Fall  Year 1  Spring
First Year Only
CS1000 (1) Explorations in Computing
MA1031/1032 (Concurrently)
CS1121 (3)/
CS1131 (5)
Intro Prog.
CS2111/1131
Discret Structures
CS2311 (3)
Data Structures
MA2160 (4)
Calculus II
MA1160/1161
Calc II
MA1031/1032

Fall  Year 2  Spring
MA1160/1161
CS1112/1131
CS1141 (2)
C for JAVA prog.
CS425 (3)
Databases
MA2720 (4)/
MA3710 (3)
Statistics
MA2160 (4)
Calculus II
MA2310 (3)
Linear Algebra

Year 3
CS1122/1131
CS3421 (4)
Comp Org
MA1160/1161
CS3141 (3)
Team Software Proj
CS3311 (3)
Formal Models of Comp
CS3141
CS3411 (3)
Systems Programming
UN1015, UN1025
SS Elective (3) Dept. Required [b]
Technical Elective (3) [f]
UN1015, UN1025
Technical Elective (3) [f]
UN1015, UN1025

Year 4
CS3421 (3) Intro to Algorithms
CS3231, CS3231
CS3141
CS4710 (3) SW Qual. Assurance
CS3141
CS4760 (3) Human-Comp.
Interface
CS3471 (3) SW Proc and Mgmt

Enterprise Series ENT3950, ENT3960, ENT4950, ENT4960
Software Development Fundamentals (43 hours)

- Reading and writing programs in multiple programming languages
- Utilize modern development and testing tools
- Focuses on the entire software development process as well
- Includes:
  - Algorithms and design
  - Data structures
Discrete structures (41 hours)

- Foundational material: supports other areas
- Ability to create and understand a proof formal specification, verification, databases, cryptography
- Graph theory used in networks, operating systems, and compilers
- Logic, counting, discrete probability
Software engineering (28 hours)

- Software engineering is the discipline concerned with the application of theory, knowledge, and practice to effectively build reliable software systems that satisfy the requirements of customers and users.
- Producing software systems: professionalism, quality, schedule, and cost are critical.
- A wide variety of software engineering practices have been developed.
- Consider trade-offs when selecting and applying different practices.
Algorithms and complexity (28 hours)

- Good algorithm design is crucial for the performance of all software systems
- There are a range of algorithms that address an important set of well-defined problems
- Recognize their strengths and weaknesses, and their suitability in particular contexts
• Programming languages are the medium through which programmers precisely describe concepts, formulate algorithms, and reason about solutions

• Making informed design choices by understanding the languages supporting multiple complementary approaches

• Basic knowledge of programming language translation
The underlying hardware and software infrastructure upon which applications are constructed is collectively described by the term “computer systems”.

Broadly spans:
- Operating systems
- Parallel and distributed systems
- Communication networks
- Computer architecture
Architecture and organization (16 hours)

- Understand the hardware environment upon which all computing is based, and the interface it provides to higher software layers.
- Develop programs that can achieve high performance through a programmer's awareness of parallelism and latency.
- Select a system use through an understanding of the trade-off among various components, such as CPU clock speed, cycles per instruction, memory size, and average memory access time.
Social issues and professional practice (16 hours)

- In addition to the technical issues in computing, students must be exposed to the larger societal context of computing.
- Developing an understanding of the relevant social, ethical, legal, and professional issues.
- Anticipate the impact of introducing a given product into a given environment:
  - Enhance or degrade the quality of life.
  - Impact upon individuals, groups, and institutions?
- Legal rights of software and hardware vendors and users, ethical values.
An operating system (O/S)
- Defines an abstraction of hardware
- Manages resource sharing among the computer's users

Basic topics taught
- Interface of an operating system to networks
- Kernel and user modes
- Approaches to O/S design and implementation
Parallel and distributed computing (15 hours)

- Was a largely elective topic before multi-core processors and distributed data centers
- Logically simultaneous execution of multiple processes whose operations have the potential to interleave in complex ways
- Models of communication and coordination among processes
- Security and fault tolerance in distributed systems
Summary

- Information about me (done)
- Tips to connect with faculty (done)
- Course information (done)
- Computer Science curricula (done)

- Reliability
- Correctness
- Performance
- Abstraction
- Layers
- Trade-offs
- Representation
- Algorithms