Software as an Engineering Discipline
Software engineering

- The economies of ALL developed nations are dependent on software
- More and more systems are software controlled
- Software engineering is concerned with theories, methods and tools for professional software development
- Software engineering expenditure represents a significant fraction of GNP in all developed countries
What is the difference between software engineering and computer science?

- Computer science is concerned with theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.

- Computer science theories are currently insufficient to act as a complete underpinning for software engineering.
What is the difference between software engineering and system engineering?

• System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering. Software engineering is part of this process.

• System engineers are involved in system specification, architectural design, integration and deployment.
# Programming vs. Engineering

<table>
<thead>
<tr>
<th>Small project</th>
<th>Huge project</th>
</tr>
</thead>
<tbody>
<tr>
<td>You</td>
<td>Teams</td>
</tr>
<tr>
<td>Build what you want</td>
<td>Build what they want</td>
</tr>
<tr>
<td>One product</td>
<td>Family of products</td>
</tr>
<tr>
<td>Few sequential changes</td>
<td>Many parallel changes</td>
</tr>
<tr>
<td>Short-lived</td>
<td>Long-lived</td>
</tr>
<tr>
<td>Cheap</td>
<td>Costly</td>
</tr>
<tr>
<td>Small consequences</td>
<td>Large consequences</td>
</tr>
</tbody>
</table>

These courseware materials courtesy of Andre van der Hoeke, Copyright ©2001, André van der Hoek
From Programming to Engineering

• People
  – Who else would do the work?
  – Range from novice to very experienced

• Processes
  – To organize and manage the efforts of individuals
  – Range from informal to very formal
  – Deal with all aspects of producing software, not just coding

• Tools
  – To support the people and the processes
  – Range from simple to very advanced

People + Processes + Tools ⇒ Product
People, Processes, Tools, Products

• Products are always the eventual goal
  – Selling products creates revenue
  – Selling **good** products creates lots of revenue
  – Selling **bad** products creates little revenue

• People, processes, and tools are retained by an organization
  – Build a reputation through the quality of products
  – Create organizational culture
  – Important to keep the team intact

*People + Processes + Tools ⇒ Product*
People

• The single most important factor in the success/failure of a product
• Scarce resource
  – Quality
  – Suitability
  – Cost
• Many different kinds of people
  – Managers
  – Programmers
  – Technical writers
  – Requirements Analysts
  – Quality Assurance Engineers

These courseware materials courtesy of Andre van der Hoeke, Copyright ©2001, André van der Hoek
Process

• Institute processes through which software is engineered
  – Cover all steps from initial idea and requirements to delivery, maintenance, and final retirement
  – Make sure we do the right things/things right
  – Make sure we do not forget to do anything
  – Different processes for different kinds of software

• Not a silver bullet [Brooks “No Silver Bullet”]
  – Software is still intrinsically difficult to deal with
  – Processes help, but cannot guarantee anything

People + Processes + Tools ⇒ Product
Tools

• Needed to support people and processes
• Scarce resource
  – Quality
  – Suitability
  – Cost
• Many different kinds of tools
  – Visual Modeling
  – Analysis
  – Project management & Cost Estimation
  – Source code management
Product

• The software should deliver the required functionality and performance to the user in a cost-effective manner

• *The “-ilities” of Software Engineering:*
  • Availability
    – Software should be reliable – not break down
  • Maintainability
    – Software must evolve to meet changing needs
  • Scalability
    – Software should “gracefully degrade” under heavy loads
  • Dependability
    – Software must be trustworthy
  • Efficiency
    – Software should not make wasteful use of system resources
  • Usability
    – Software must be usable by the users for which it was designed
Building a house vs. software

- Determining and analyzing requirements
- Producing and documenting the design
- Detailed specifications
- Identifying and designing components
- Building components
- Testing components
- Integrating components
- Making final modifications
- Continuing maintenance

- Requirements analysis and definition
- System design
- Program design
- Writing programs
- Unit testing
- Integration testing
- System testing
- System delivery
- Maintenance

These courseware materials courtesy of Sara L. Pfleeger
Visibility vs...
...Invisibility

• Software as is cannot be viewed meaningfully
  – Stack of paper
  – Set of files

• Software cannot be interpreted easily
  – How to read source code?
  – How to read a million lines of source code?
  – How to read a part of source code?

• Invisibility affects process
  – How to measure progress?
    • Is a bigger stack of paper closer to the end-result than a smaller stack of paper?
Manageable Complexity vs...
...Unmanageable Complexity

• Software cannot be easily abstracted
  – Formulas
    • Only in very few domains
  – Diagrams, graphs, and other representations
    • Typically non-hierarchical
    • Far too many cross-references
  – Few concepts are available to use in an abstraction

• Tension between high-level understanding and low-level detailed specification
  – High-level understanding leaves out important details
  – Aggregation often does not work
Environment Can Be Changed vs...
...Environment Cannot Be Changed

• Software has to adhere to the “world” it is placed in
  – Cannot change the hardware
  – Cannot change the way people do business
• The “world” is often not clearly specified
  – How can you change something that you cannot specify?
  – Leads to many software changes
• Perception is that software is easier to change
No Major Changes vs...

These courseware materials courtesy of Andre van der Hoeke, Copyright ©2001, André van der Hoek
...Major Changes

• Software is remarkably easy to change
  – Change the source code, recompile, rerun
  – “One line here, one line there”
• Unfortunately, even small changes can have disastrous consequences
  – A single wrong character can surreptitiously change the behavior of the software
    • The effects of most changes are only visible in certain circumstances
• Sometimes, the environment does change
  – Software is used in different organizations
  – Software is used for different purposes
Drastic Consequences

• Deceased patients
  – X-ray machine delivered very high doses because of a timing problem in its control software

• Crashed planes
  – Software prevented pilots from performing emergency maneuvers
  – Software had similar codes for different airports

• Decreased national security
  – NSA computers down for four days due to a “software problem”
Summary

• New graduates seeking employment typically want a title “software engineer” not “computer scientist”
• Software engineering is the process and practices of engineering applied to creating software products
• Software products have unique properties that impact these processes and practices
• Next week we will discuss various software lifecycle process models – the process part
• The course will cover practices in a specific aspect of those processes, namely requirements engineering