Instructor

- Narayan Ganesan, Office: Burchard 412

- **Research Interests:** High Performance Computing, GPU/FPGA based Hybrid design, Mathematical Modeling and systems theory, Computational Biology, Systems biology.

- **Office Hours:** Tu: 10-12am

- **Email:** nganesan@stevens.edu

- **CPE640 URL:** http://personal.stevens.edu/~nganesan/CPE640/CPE640.html
Software Engineering – CPE 640

- A course on Software Creation, organization and Design
- Combines all the vital aspects of modern/usable software creation
  - **Three courses in one**
    - Crash course on Object Oriented Design Principles, Design Patterns
    - Course on Software process, modeling, life cycle and design
    - Course on practical team creation of Software for real-world problems
- Object Oriented: All real world processes and interactions can be and should be modeled as Objects
- Software Process Engineering/Modeling: Needed in order to produce “quality software” not just working programs.
- Team Work and Software Projects: Implement the strategies learnt in class. Concepts can be remembered only if they are used in real world.
Course Requirements:

- One lecture per week: Tu 6:15-8:45pm
- 6 Homeworks (50%)
- Project (50%)
  - Project Proposal (20%): Propose Projects in teams
  - Project Demo (20%): This is a teamwork
  - Project Documentation and report submission (10%): Good writing skills and professionalism is expected and will be graded upon.
- Need to work in teams of 3-4
- TA yet to be announced
- Project topics will be suggested discussed in class, but you have the freedom to propose, choose and design your own project
Several textbooks are available for software engineering, (hundreds...), you are welcome to refer to any of those for further information. Few of the books are listed...

Most textbook focus on design principles, software life cycle and methods, general practices, and theory...

Internet is a rich warehouse for software engineering tools and techniques.

Lecture notes also outline and summarize the design techniques.

However, in this course we follow a textbook meant primarily to model software in UML...

UML specifications are used in real workplace and is a very concise modeling tool to present software features, functions and design...
Textbook:

Design Patterns: Elements of Reusable Object-Oriented Software, by Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides

Optional Textbook(s)

Head First Design Patterns, by Elisabeth Freeman, Eric Freeman, Bert Bates, Kathy Sierra, Elisabeth Robson

Object-Oriented Software Engineering Using UML, Patterns, and Java (3rd Edition) Bernd Bruegge (Author), Allen H. Dutoit (Author)
Introduction to Software Engineering
Chapter 1- Introduction

Lecture 1
Topics covered

- Professional software development
  - What is meant by software engineering.
- Software engineering ethics
  - A brief introduction to ethical issues that affect software engineering.
- Case studies
  - An introduction to three examples that are used in later chapters in the book.
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.
- Expenditure on software represents a significant fraction of GNP in all developed countries.
Software costs

- Software costs often dominate computer system costs. The costs of software on a PC are often greater than the hardware cost.

- Software costs more to maintain than it does to develop. For systems with a long life, maintenance costs may be several times development costs.

- Software engineering is concerned with cost-effective software development.
Software products

- **Generic products**
  - Stand-alone systems that are marketed and sold to any customer who wishes to buy them.
  - Examples – PC software such as graphics programs, project management tools; CAD software; software for specific markets such as appointments systems for dentists.

- **Customized products**
  - Software that is commissioned by a specific customer to meet their own needs.
  - Examples – embedded control systems, air traffic control software, traffic monitoring systems.
Product specification

- Generic products
  - The specification of what the software should do is owned by the software developer and decisions on software change are made by the developer.

- Customized products
  - The specification of what the software should do is owned by the customer for the software and they make decisions on software changes that are required.
Why Software Engineering Matters

Software’s contribution to US economy (1996 figures):

- greatest trade surplus of exports
- $24B software exported, $4B imported, $20B surplus
- compare: agriculture 26-14-12, aerospace 11-3-8, chemicals 26-19-7, vehicles 21-43-(22), manufactured goods 200-265-(64)


Role in infrastructure:

- Not just the Internet
- Transportation, energy, medicine, finance
- Software is becoming pervasive in embedded devices. New cars, for example, have between 10 and 100 processors for managing all kinds of functions from music to braking
Cost of acquiring and maintaining software

- Cost of software:
  - Ratio of hardware to software procurement cost approaches zero
  - Total cost of ownership: 5 times cost of hardware. Gartner group estimates cost of keeping a PC for 5 years is now $7-14k
State of Software development and design...

How good is our software?

- Failed developments
- Accidents
- Poor quality software

**Cancelled Software Projects**: Development of large applications in excess of 5000 function points, (~500,000 LOC) is the most risky business undertaking in the modern economy (Capers Jones). Research by: **Capers Jones, Software Productivity Research Inc**

- Risks of cancellation or major delays rise rapidly as overall application size increases (Capers Jones):
  - 25% for those over 100,000 LOC
  - 50% for systems exceeding half million LOC
  - 65% of large systems (over 1,000,000 LOC) are cancelled before completion
- Failure or cancellation rate of large software systems is over 20% (Capers Jones)
Software Project Cancellation Nightmares…

- After surveying 8,000 IT projects, Standish Group reported about 30% of all projects were cancelled.

- Average cancelled project in U.S. is about a year behind schedule and has consumed 200% of expected budget (Capers Jones).

- Work on cancelled projects comprises about 15% of total U.S. software efforts, amounting to as much as $14 billion in 1993 dollars (Capers Jones).

- Of completed projects, 2/3 experience schedule delays and cost overruns (Capers Jones) [bad estimates?] 

- 2/3 of completed projects experience low reliability and quality problems in first year of deployment (Jones).

- Software errors in fielded systems typically range from 0.5 to 3.0 occurrences per 1000 lines of code (Bell Labs survey).

- Civilian software: at least 100 English words produced for every source code statement. Military: about 400 words (Capers Jones)
## Essential attributes of good software

<table>
<thead>
<tr>
<th>Product characteristic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maintainability</td>
<td>Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.</td>
</tr>
<tr>
<td>Dependability and security</td>
<td>Software dependability includes a range of characteristics including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.</td>
</tr>
<tr>
<td>Efficiency</td>
<td>Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilisation, etc.</td>
</tr>
<tr>
<td>Acceptability</td>
<td>Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use.</td>
</tr>
</tbody>
</table>
Software engineering is an engineering discipline that is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use.

Engineering discipline

- Using appropriate theories and methods to solve problems bearing in mind organizational and financial constraints.

All aspects of software production

- Not just technical process of development. Also project management and the development of tools, methods etc. to support software production.
Importance of software engineering

- More and more, individuals and society rely on advanced software systems. We need to be able to produce reliable and trustworthy systems economically and quickly.

- It is usually cheaper, in the long run, to use software engineering methods and techniques for software systems rather than just write the programs as if it was a personal programming project. For most types of system, the majority of costs are the costs of changing the software after it has gone into use.
### Frequently asked questions about software engineering

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>What is software?</strong></td>
<td>Computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market.</td>
</tr>
<tr>
<td><strong>What are the attributes of good software?</strong></td>
<td>Good software should deliver the required functionality and performance to the user and should be maintainable, dependable and usable.</td>
</tr>
<tr>
<td><strong>What is software engineering?</strong></td>
<td>Software engineering is an engineering discipline that is concerned with all aspects of software production.</td>
</tr>
<tr>
<td><strong>What are the fundamental software engineering activities?</strong></td>
<td>Software specification, software development, software validation and software evolution.</td>
</tr>
<tr>
<td><strong>What is the difference between software engineering and computer science?</strong></td>
<td>Computer science focuses on theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.</td>
</tr>
<tr>
<td><strong>What is the difference between software engineering and system engineering?</strong></td>
<td>System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering. Software engineering is part of this more general process.</td>
</tr>
</tbody>
</table>
# Frequently asked questions about software engineering

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>What are the key challenges facing software engineering?</td>
<td>Coping with increasing diversity, demands for reduced delivery times and developing trustworthy software.</td>
</tr>
<tr>
<td>What are the costs of software engineering?</td>
<td>Roughly 60% of software costs are development costs, 40% are testing costs. For custom software, evolution costs often exceed development costs.</td>
</tr>
<tr>
<td>What are the best software engineering techniques and methods?</td>
<td>While all software projects have to be professionally managed and developed, different techniques are appropriate for different types of system. For example, games should always be developed using a series of prototypes whereas safety critical control systems require a complete and analyzable specification to be developed. You can't, therefore, say that one method is better than another.</td>
</tr>
<tr>
<td>What differences has the web made to software engineering?</td>
<td>The web has led to the availability of software services and the possibility of developing highly distributed service-based systems. Web-based systems development has led to important advances in programming languages and software reuse.</td>
</tr>
</tbody>
</table>
Software process activities

- Software specification, where customers and engineers define the software that is to be produced and the constraints on its operation.

- Software development, where the software is designed and programmed.

- Software validation, where the software is checked to ensure that it is what the customer requires.

- Software evolution, where the software is modified to reflect changing customer and market requirements.
General issues that affect most software

- **Heterogeneity**
  - Increasingly, systems are required to operate as distributed systems across networks that include different types of computer and mobile devices.

- **Business and social change**
  - Business and society are changing incredibly quickly as emerging economies develop and new technologies become available. They need to be able to change their existing software and to rapidly develop new software.

- **Security and trust**
  - As software is intertwined with all aspects of our lives, it is essential that we can trust that software.
There are many different types of software system and there is no universal set of software techniques that is applicable to all of these.

The software engineering methods and tools used depend on the type of application being developed, the requirements of the customer and the background of the development team.
Application types

- **Stand-alone applications**
  - These are application systems that run on a local computer, such as a PC. They include all necessary functionality and do not need to be connected to a network.

- **Interactive transaction-based applications**
  - Applications that execute on a remote computer and are accessed by users from their own PCs or terminals. These include web applications such as e-commerce applications.

- **Embedded control systems**
  - These are software control systems that control and manage hardware devices. Numerically, there are probably more embedded systems than any other type of system.
Application types

- **Batch processing systems**: Business, Enterprise, financial system, engineering, medical, etc.
  - These are business systems that are designed to process data in large batches. They process large numbers of individual inputs to create corresponding outputs.

- **Entertainment systems**: Games and Multimedia
  - These are systems that are primarily for personal use and which are intended to entertain the user.

- **Systems for modelling and simulation**: Scientific Software
  - These are systems that are developed by scientists and engineers to model physical processes or situations, which include many, separate, interacting objects.
Application types

- **Data collection systems**: sensor fusion, drivers, embedded software
  - These are systems that collect data from their environment using a set of sensors and send that data to other systems for processing.

- **Systems of systems**: Heterogeneous software systems
  - These are systems that are composed of a number of other software systems.
Some fundamental principles apply to all types of software system, irrespective of the development techniques used:

- Systems should be developed using a managed and understood development process. Of course, different processes are used for different types of software.
- Dependability and performance are important for all types of system.
- Understanding and managing the software specification and requirements (what the software should do) are important.
- Where appropriate, you should reuse software that has already been developed rather than write new software.
What is software engineering?

- Software engineering is an engineering discipline that is concerned with all aspects of software production.
- Software engineers should adopt a systematic and organised approach to their work and use appropriate tools and techniques depending on the problem to be solved, the development constraints and the resources available.
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 still insufficient to act as a complete underpinning for software engineering (unlike e.g. physics and electrical 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 concerned with developing the software infrastructure, control, applications and databases in the system.

- System engineers are involved in system specification, architectural design, integration and deployment.
What is a software process?

- A set of activities whose goal is the development or evolution of software.

- Generic activities in all software processes are:
  - Specification - what the system should do and its development constraints
  - Development - production of the software system
  - Validation - checking that the software is what the customer wants
  - Evolution - changing the software in response to changing demands.
What is a software process model?

- A simplified representation of a software process, presented from a specific perspective.

- Examples of process perspectives are:
  - Workflow perspective - sequence of activities;
  - Data-flow perspective - information flow;
  - Role/action perspective - who does what.

- Generic process models:
  - Waterfall;
  - Iterative development;
  - Component-based software engineering.
What are the costs of software engineering?

- Roughly 60% of costs are development costs, 40% are testing costs. For custom software, evolution costs often exceed development costs.

- Costs vary depending on the type of system being developed and the requirements of system attributes such as performance and system reliability.

- Distribution of costs depends on the development model that is used.
Activity cost distribution

- **Waterfall model**
  - Specification: 0%
  - Design: 25%
  - Development: 50%
  - Integration and testing: 75%

- **Iterative development**
  - Specification: 0%
  - Iterative development: 25%
  - System testing: 75%

- **Component-based software engineering**
  - Specification: 0%
  - Development: 25%
  - Integration and testing: 75%

- **Development and evolution costs for long-lifetime systems**
  - System development: 10%
  - System evolution: 30%

Post development costs a.k.a Maintenance

On an Average:

35% planning
20% coding

45% Testing which is further divided into:
  25% Component Testing
  20% System Testing
Software Maintenance

Software Maintenance:
20% error correction
20% adaptation
60% enhancements

Development costs are only the tip of the iceberg
IBM survey, 1994
- 55% of systems cost more than expected
- 68% overran schedules
- 88% had to be substantially redesigned

Advanced Automation System (FAA, 1982-1994)
- Industry average was $100/line, expected to pay $500/line
- Ended up paying $700-900/line
- $6B worth of work discarded

- For every 6 new systems put into operation, 2 cancelled
- Probability of cancellation is about 50% for biggest systems
- Average project overshoots schedule by 50%
- 3/4 systems are regarded as ‘operating failures’
Radiotherapy machine with software controller. Therac-25 (1985-87)

- hardware interlock removed, but software had no interlock
- Software failed to maintain essential invariants: either electron beam mode or stronger beam and plate intervening, to generate X-rays
- several deaths due to burning
- programmer had no experience with concurrent programming
- see: [http://sunnyday.mit.edu/therac-25.html](http://sunnyday.mit.edu/therac-25.html)

Another Incident:

- International Atomic Energy Agency declared ‘radiological emergency’ in Panama on 22 May, 2001
- 28 patients overexposed; 8 died, of which 3 as result; 3/4 of surviving 20 expected to develop ‘serious complications which in some cases may ultimately prove fatal’
- Experts found radiotherapy equipment ‘working properly’; cause of emergency lay with data entry
- If data entered for several shielding blocks in one batch, incorrect dose computed. FDA, at least, concluded that ‘interpretation of beam block data by software’ was a factor

- see [http://www.fda.gov/cdrh/ocd/panamaradexp.html](http://www.fda.gov/cdrh/ocd/panamaradexp.html)
Ariane-5 (June 1996)

- European Space Agency
- Complete loss of unmanned rocket shortly after takeoff
- Due to exception thrown in Ada code
- Faulty code was not even needed after takeoff
- Due to change in physical environment: undocumented assumptions violated

http://www.esa.int/htdocs/tidc/Press/Press96/ariane5rep.html

- Failure to articulate and evaluate requirements properly, environmental interactions.
Software Failures

  - loss of calls, double dispatches from duplicate calls
  - poor choice of developer: inadequate experience
  - see: [http://www.cs.ucl.ac.uk/staff/A.Finkelstein/las.html](http://www.cs.ucl.ac.uk/staff/A.Finkelstein/las.html)

- Inadequate testing and validation, improper planning leads to failure of fielded software.
Software failures...

- In the short term, these problems will become worse because of the pervasive use of software in our civic infrastructure.
- PITAC report recognized this, and has successfully argued for increase in funding for software research:
  “The demand for software has grown far faster than our ability to produce it. Furthermore, the Nation needs software that is far more usable, reliable, and powerful than what is being produced today. We have become dangerously dependent on large software systems whose behavior is not well understood and which often fail in unpredicted ways.”

President’s Information Technology Advisory Committee (PITAC), Report to the President, February 24, 1999

RISKS Forum

collates reports from press of computer-related incidents

http://catless.ncl.ac.uk

Such dependence on software, calls for strong software engineering design methods that is based on sound principles in order to produce quality software.
What are software engineering methods?

- Structured approaches to software development which include system models, notations, rules, design advice and process guidance.

- Model descriptions
  - Descriptions of graphical models which should be produced;

- Rules
  - Constraints applied to system models;

- Recommendations
  - Advice on good design practice;

- Process guidance
  - What activities to follow.
Software for the Web

Web-based software design strategies and properties.
Software engineering and the web

- The Web is now a platform for running application and organizations are increasingly developing web-based systems rather than local systems.

- Web services allow application functionality to be accessed over the web.

- Cloud computing is an approach to the provision of computer services where applications run remotely on the ‘cloud’.
  - Users do not buy software but pay according to use.
Software reuse is the dominant approach for constructing web-based systems.

When building these systems, you think about how you can assemble them from pre-existing software components and systems.

Web-based systems should be developed and delivered incrementally.

It is now generally recognized that it is impractical to specify all the requirements for such systems in advance.

User interfaces are constrained by the capabilities of web browsers.

Technologies such as AJAX allow rich interfaces to be created within a web browser but are still difficult to use. Web forms with local scripting are more commonly used.
Web-based software engineering

- Web-based systems are complex distributed systems but the fundamental principles of software engineering discussed previously are as applicable to them as they are to any other types of system.

- The fundamental ideas of software engineering, discussed in the previous section, apply to web-based software in the same way that they apply to other types of software system.
Key points

- Software engineering is an engineering discipline that is concerned with all aspects of software production.

- Essential software product attributes are maintainability, dependability and security, efficiency and acceptability.

- The high-level activities of specification, development, validation and evolution are part of all software processes.

- The fundamental notions of software engineering are universally applicable to all types of system development.
Key points

- There are many different types of system and each requires appropriate software engineering tools and techniques for their development.

- The fundamental ideas of software engineering are applicable to all types of software system.
Code of Ethics as chartered by IEEE/ACM requires all professional software engineers to read and follow the code in their day-to-day professional activities.
Software engineering ethics

- Software engineering involves wider responsibilities than simply the application of technical skills.

- Software engineers must behave in an honest and ethically responsible way if they are to be respected as professionals.

- Ethical behaviour is more than simply upholding the law but involves following a set of principles that are morally correct.
Issues of professional responsibility

- **Confidentiality**
  - Engineers should normally respect the confidentiality of their employers or clients irrespective of whether or not a formal confidentiality agreement has been signed.

- **Competence**
  - Engineers should not misrepresent their level of competence. They should not knowingly accept work which is outwith their competence.
Issues of professional responsibility

- **Intellectual property rights**
  - Engineers should be aware of local laws governing the use of intellectual property such as patents, copyright, etc. They should be careful to ensure that the intellectual property of employers and clients is protected.

- **Computer misuse**
  - Software engineers should not use their technical skills to misuse other people’s computers. Computer misuse ranges from relatively trivial (game playing on an employer’s machine, say) to extremely serious (dissemination of viruses).
The professional societies in the US have cooperated to produce a code of ethical practice.

Members of these organisations sign up to the code of practice when they join.

The Code contains eight Principles related to the behaviour of and decisions made by professional software engineers, including practitioners, educators, managers, supervisors and policy makers, as well as trainees and students of the profession.
Rationale for the code of ethics

- Computers have a central and growing role in commerce, industry, government, medicine, education, entertainment and society at large. Software engineers are those who contribute by direct participation or by teaching, to the analysis, specification, design, development, certification, maintenance and testing of software systems.

- Because of their roles in developing software systems, software engineers have significant opportunities to do good or cause harm, to enable others to do good or cause harm, or to influence others to do good or cause harm. To ensure, as much as possible, that their efforts will be used for good, software engineers must commit themselves to making software engineering a beneficial and respected profession.
The ACM/IEEE Code of Ethics

Software Engineering Code of Ethics and Professional Practice

ACM/IEEE-CS Joint Task Force on Software Engineering Ethics and Professional Practices

PREAMBLE
The short version of the code summarizes aspirations at a high level of the abstraction; the clauses that are included in the full version give examples and details of how these aspirations change the way we act as software engineering professionals. Without the aspirations, the details can become legalistic and tedious; without the details, the aspirations can become high sounding but empty; together, the aspirations and the details form a cohesive code.

Software engineers shall commit themselves to making the analysis, specification, design, development, testing and maintenance of software a beneficial and respected profession. In accordance with their commitment to the health, safety and welfare of the public, software engineers shall adhere to the following Eight Principles:
1. PUBLIC - Software engineers shall act consistently with the public interest.

2. CLIENT AND EMPLOYER - Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.

3. PRODUCT - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.

4. JUDGMENT - Software engineers shall maintain integrity and independence in their professional judgment.

5. MANAGEMENT - Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.

6. PROFESSION - Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.

7. COLLEAGUES - Software engineers shall be fair to and supportive of their colleagues.

8. SELF - Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.
Ethical dilemmas

- Disagreement in principle with the policies of senior management.
- Your employer acts in an unethical way and releases a safety-critical system without finishing the testing of the system.
- Participation in the development of military weapons systems or nuclear systems.
Projects will be discussed in the next class. The students are free to choose their own projects or consult me.

Form a group of 3-4 students per team as soon as possible. Efforts of each student in the group must be disclosed and the effort will affect the grading.

Suggested Topics for projects:

- Web based: Java, SQL based projects
- Stand-alone: Python, Java, C++ or choice of your OOPL.
- Social media and Social Networks: Develop a facebook app, twitter app or an Opensocial application.
- Other Web-based APIs: Google maps API to monitor and suggest real time traffic updates and routes.
What should the report contain:

The project report should contain detailed description of the following:

- Project description and requirements elicitation
- Design in UML and software process modeling (to be studied in the next class)
- Cost and effort Estimation analysis
- Implementation in language and platform of your choice or as required by the project.
- Validation and Testing: choice of test cases and why
- Documentation and plans for handover in case of real client.