

SENG 265:

Software Development Methods

Spring 2013



SENG 265

- **Software Development Methods**
- Instructor: Michael Zastre
 - ECS 528
 - zastre@cs.uvic.ca
 - Office hours: Wednesdays 12:30 to 1:30;
Thursdays 11:30 to 1:30; or by appointment
- Lab instructor: Sonmaz Zehtabi
 - Labs begin week of January 7th
 - Engineering Lab Wing B215

Administrative Details

- Course web page
 - Via “connex.csc.uvic.ca”
 - Course appears as a tab when you log into `conneX`
- Lab sections
 - Focus is on hands-on + tutorial components
 - You must register for a lab section
 - Attendance at labs is mandatory

Your course account

- The details below (and more) will be covered in the first lab session
- Use your Netlink credentials
- You can remotely log in to any of the lab machines or into the server
- These machines all run Linux
- For more information on SENG labs, visit:
 - <http://labs.seng.engr.uvic.ca>
- If you do not have a CSC account (needed for conneX access):
 - <http://accounts.csc.uvic.ca>



Grading

- Breakdown:
 - assignments: 32% (4 assignments @ 8%)
 - lab work: 10% (10 labs)
 - midterm exam: 18%
 - final exam: 40%
- Marking disputes (“one-week rule”)
- Midterm: February 12th (Tuesday)
- Final exam: Scheduled by University

Purpose of Course

- General introduction to:
 - software development methodologies
 - production languages
 - UNIX/Linux environment and scripting
- Preparation for upcoming workterms
- Working at a higher level of abstraction
- Acquiring and reinforcing good habits when writing software and software systems

Context

- Your experience thus far:
 - small, relatively simple programs
 - provided with steps to solving specific problem
 - written alone
 - no ongoing maintenance
- What awaits in industry:
 - large and complex projects
 - do not know ahead of time how to solve the problem
 - work in teams (often very specialized)
 - ongoing maintenance is critical

Course topics

- Software development “process”
- UNIX/Linux fundamentals
- C programming
- Python programming
- Inspection, profiling, testing and debugging
- Source code control, code revision and change management
- “Pragmatic programming”

By the end of this course...

- You should be able to:
 - program with some comfort in a UNIX environment
 - use Python for prototyping, and to support code testing and debugging
 - recognize a problem statement that can become a program specification
 - use general-purpose languages such as C and Python to solve programming problems
 - work with code versioning systems to manage changes in your own code
 - apply some general software engineering techniques to your own projects
 - be ready to delve deeper into more formal software engineering approaches

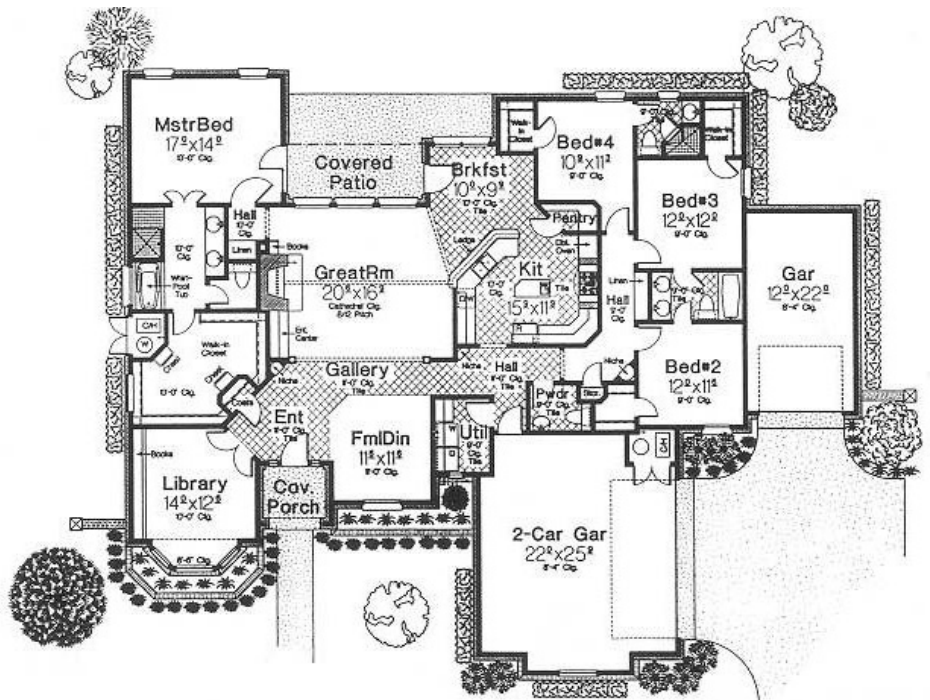
Academic integrity

- Guiding principles
 - discussion is encouraged ...
 - .. but work submitted for credit must be your own
 - in cases where attribution is appropriate, it must be given
 - example: code taken from a textbook or web-based tutorial
 - example: algorithm based on a journal paper
- Computer Science departmental guidelines:
 - <http://www.csc.uvic.ca/courseinfo/policies/fraud.html>
- If you are unclear, please ask the instructor.
- Attribution for these slides!
 - They were originally created by myself and then lightly edited by Nigel Horspool in Summer 2009, then more by me in Fall 2010, Fall 2011, etc. etc.



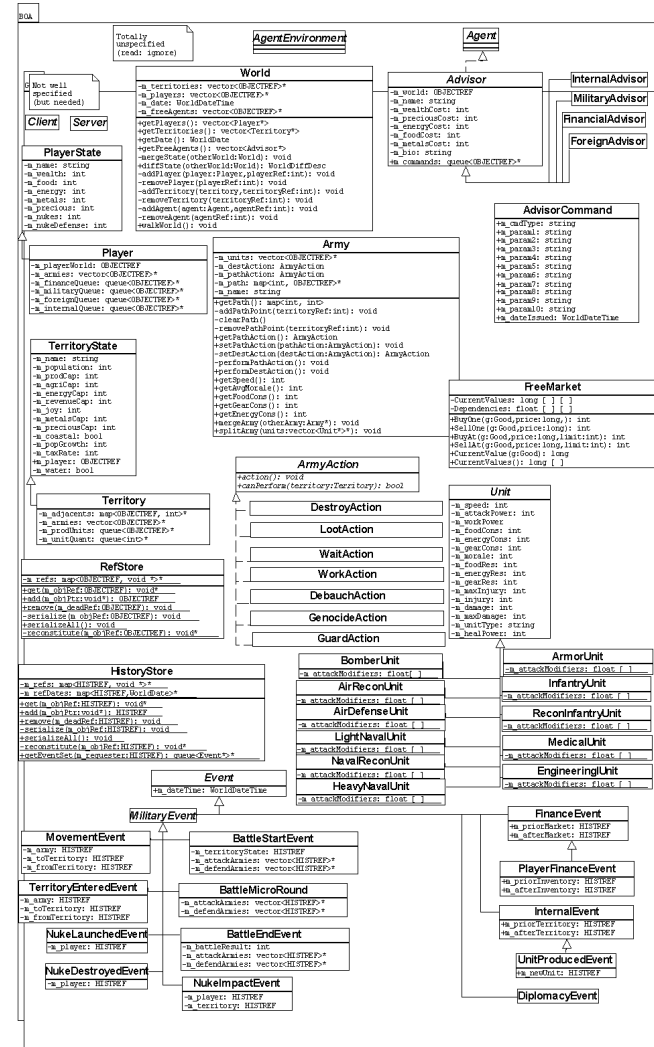
A new building: methodology

- determining and analyzing requirements
- producing and documenting overall design
- producing the detailed specifications of the house
- identifying and designing the components
- building each component
- testing/inspecting each component
- integrating the components
- making final modifications after residents have moved in
- ongoing maintenance

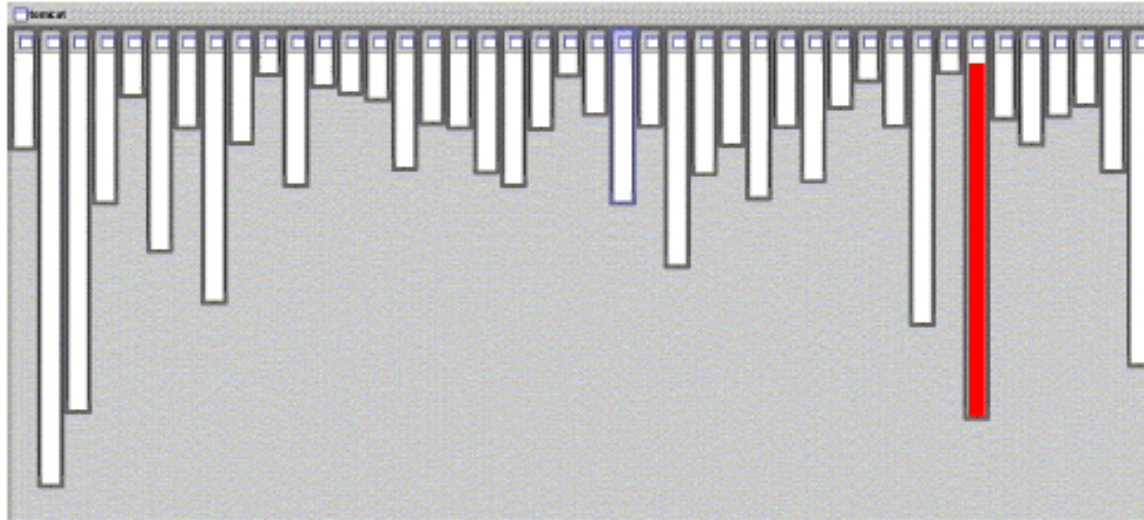


A new program: methodology

- requirements analysis and definition
- system design
- program design
- writing the programs (program implementation)
- unit testing
- integration testing
- system testing
- system delivery
- maintenance

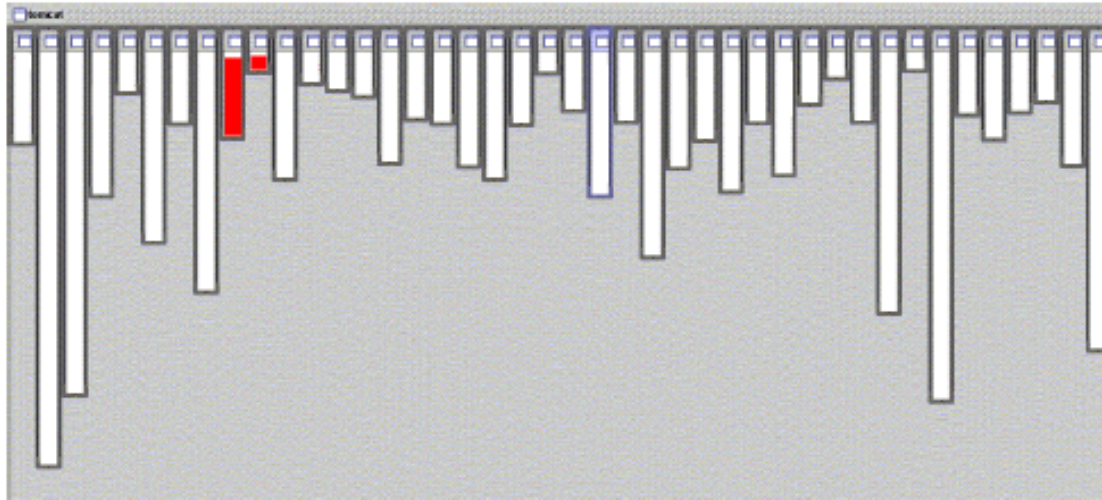


code locality in Tomcat



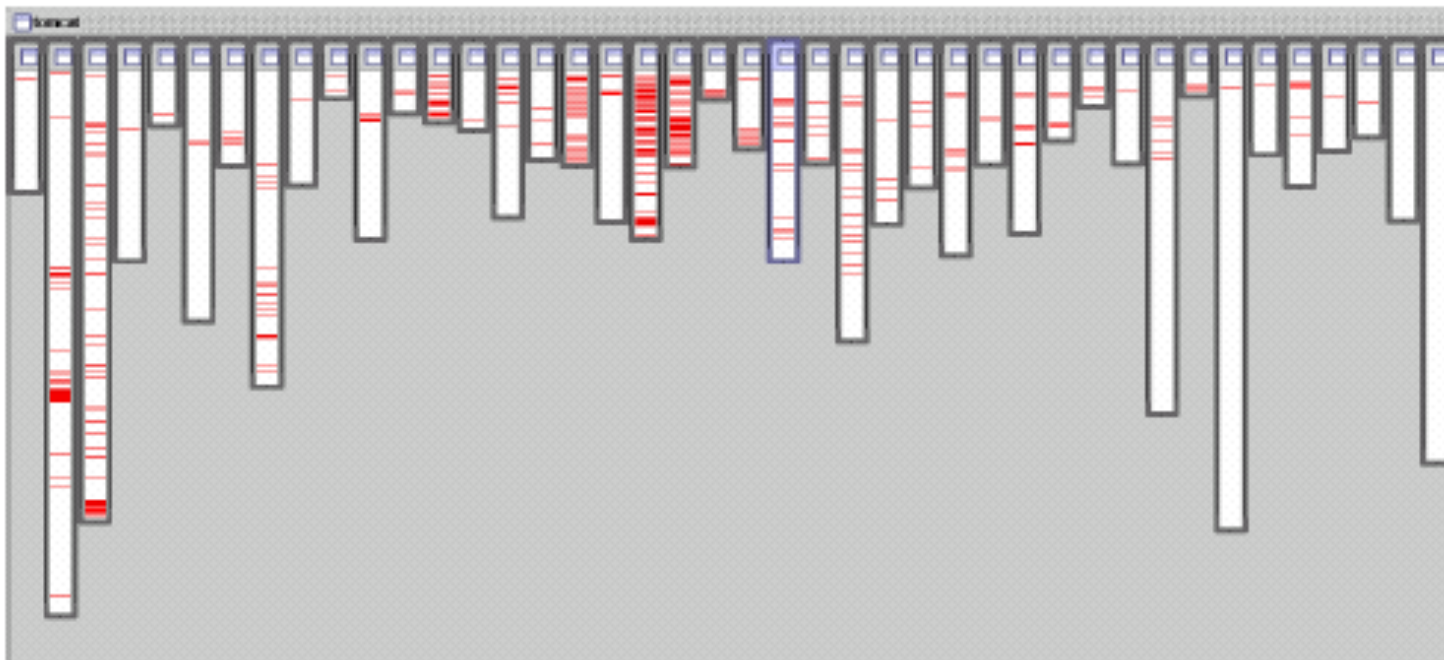
- XML parsing in org.apache.tomcat
 - each column corresponds to a class
 - length of each column indicates size of class
 - red colour represents the code lines relevant to XML parsing
 - what this shows is good modularity (i.e., all XML parsing code is in one module)

code locality in Tomcat (2)



- URL pattern matching in `org.apache.tomcat`
 - functionality is now spread over two classes
 - modularity is still good
 - any changes to URL pattern matching code restricted to these two modules

problems



- logging code in Tomcat is not well-modularized
 - again, red corresponds to lines of code devoted to logging
 - nearly every class has some logging code
 - classic example of “cross-cutting concern”
 - concern: logging
 - cross-cutting: across many classes



session expiration: also spread out



Development environment

- On the surface, there would appear to be two families of development tools:
 - those which employ a **GUI** (graphical user interface), typically as part of an **IDE** (integrated development environment)
 - those which employ a **CLI** (command line interface)
- An IDE tends to hide many of the details of the development process

Development environment (2)

- In this course we use a command-line interface;
 - this gives us a better understanding of aspects of the development process which might be hidden inside an IDE
 - compilation
 - source code management
 - profiling
 - testing, etc.
- Our command line interface ("bash") is run within the Linux variant of Unix.



Purpose of our environment choice

- Simplicity
- Universality
- Professional (sometimes more powerful) tools
- Less “mysterious automation” of programming steps
- Not intended to make your life harder:
 - absence of tools with which you are familiar is not necessarily bad
 - goal is that you should be able to make an informed choice when deciding upon your tools and environment for a given task

Next steps

- Introduction to Unix
 - Its architecture
 - Use of the shell
 - Working within the shell
- Subversion
 - Simple, one-person-oriented source code control
 - open-source tool
 - widely used, lots of support
 - (yes, I know some of you are keen on git, but we'll use Subversion this term)