COURSE INTRODUCTION

SENG 265:
Software Development Methods

Fall 2017

University of Victoria
Department of Computer Science

SENG 265

- Instructor: Daniela Damian
  - ECS 558
  - e-mail: danielad@uvic.ca or damian.daniela@gmail.com (please include "SENG 265" in subject line; your message might be missed otherwise)
  - Office hours: TBA

- Labs:
  - Begin week of September 12th
  - ECS 342

Administrative Details

- Course website:
  - Via “connex.csc.uvic.ca”
  - Course appears as a tab when you log into conneX
  - The tab might not immediately appear if you've taken several CSC courses already
- Lab sections:
  - Our focus is on hands-on + tutorial components
  - You must register for a lab section
  - Attendance at labs is highly recommended

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
- If you do not have a CSC account (needed for conneX access), then activate your account at:
  - http://accounts.csc.uvic.ca

Grading

- Breakdown:
  - assignments: 40% (4 assignments @ 10%)
  - Quizzes: 6%
  - midterm exam: 18%
  - final exam: 36%
- Marking disputes ("one-week rule")
- Midterm: October 19 (Thursday)
- Final exam: Scheduled by University
- Course outline: https://heat.csc.uvic.ca/coview/course/2017091/SENG265

Purpose of Course

- General introduction to:
  - UNIX/Linux environment and scripting
  - production languages (C & Python)
  - software development methodologies
- 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

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

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
- UVic Academic Integrity guidelines:
  - http://www.uvic.ca/current-students/home/academics/academic-integrity/index.php
- Attribution for these slides!
  - They were originally created by Mike Zastre in consultation with Nigel Horspool, and used in teaching SENG265 for many years at Uvic; adapted by me for teaching this term

Development environment

- At first glance 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
  - 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
- Git
  - open-source version control system (VCS)
  - widely used, yet with wildly varying workflows
  - (Swiss-army knife approach to VCS…)
  - we will use our own BSEng Git server (i.e., we will **not** use GitHub)

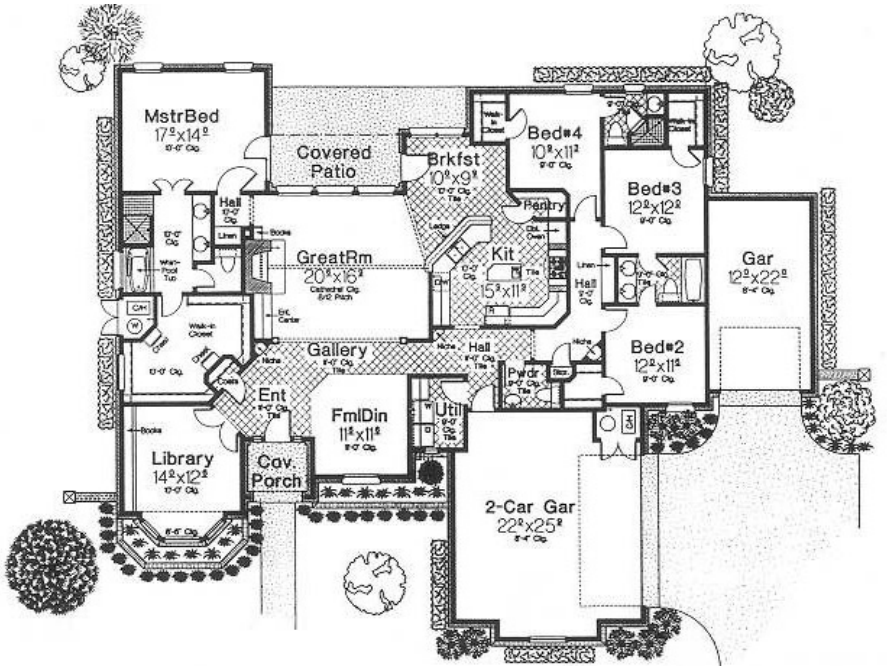
COURSE INTRODUCTION – CONT'D

SENG 265:
Software Development Methods

Fall 2017

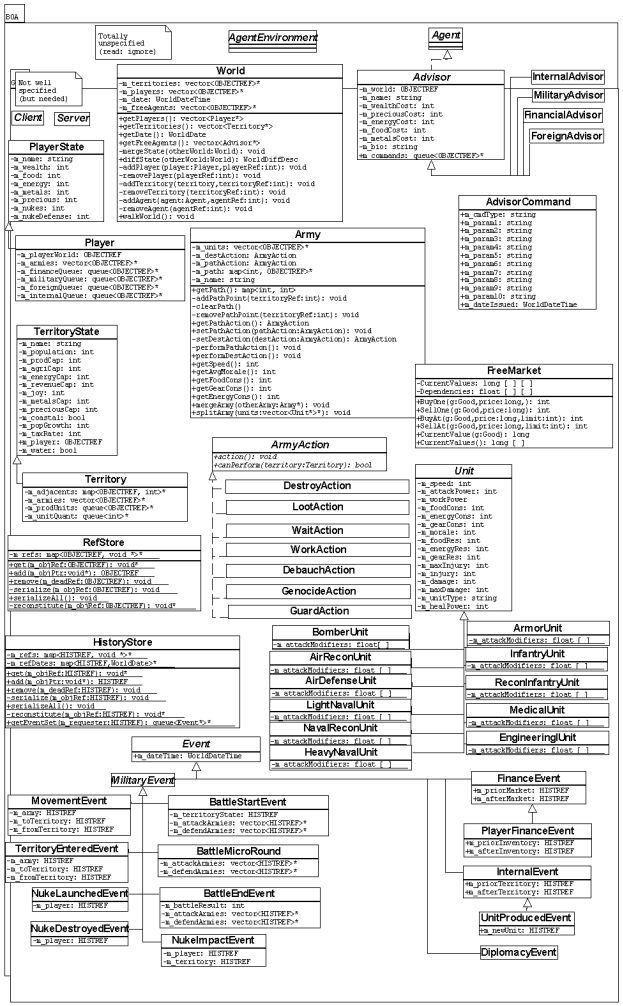
A new building: methodology

- 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**

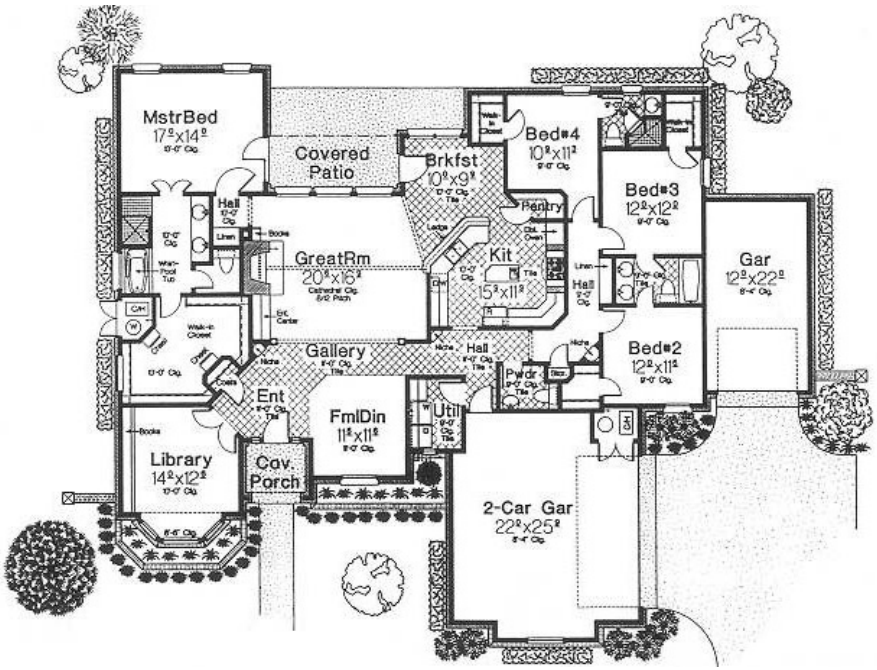


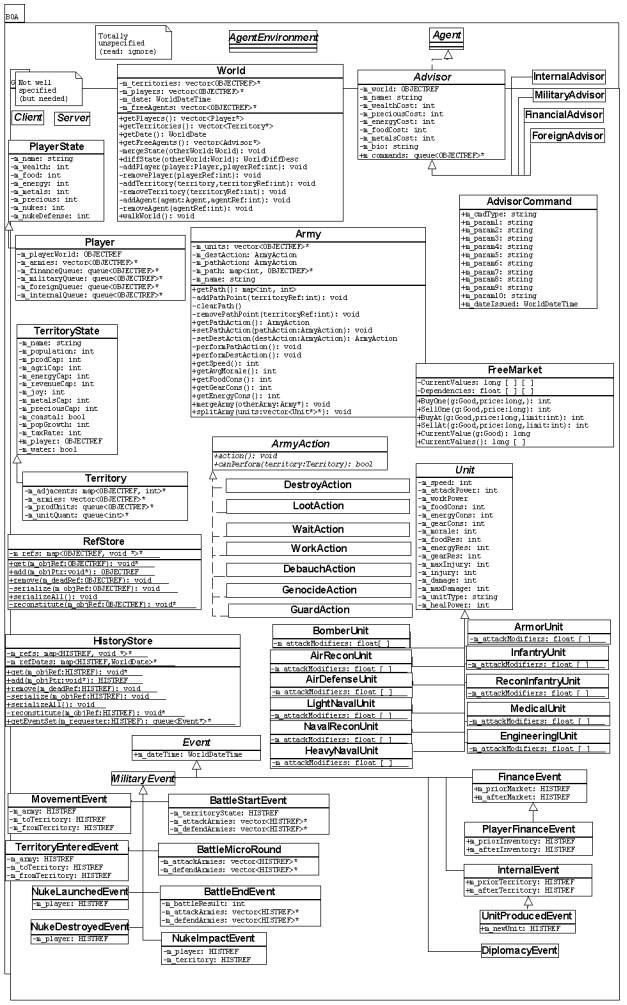
A new program: methodology

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

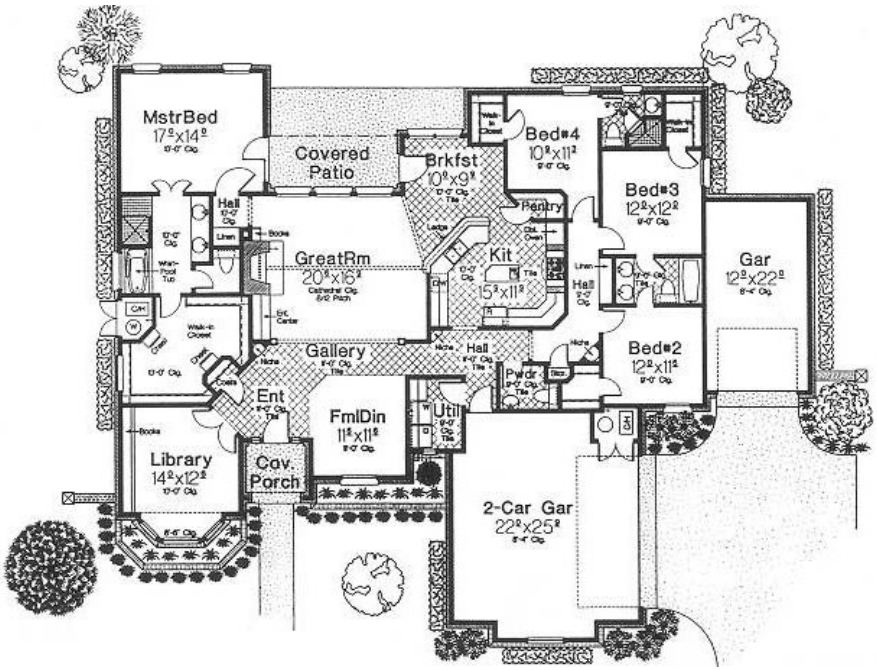


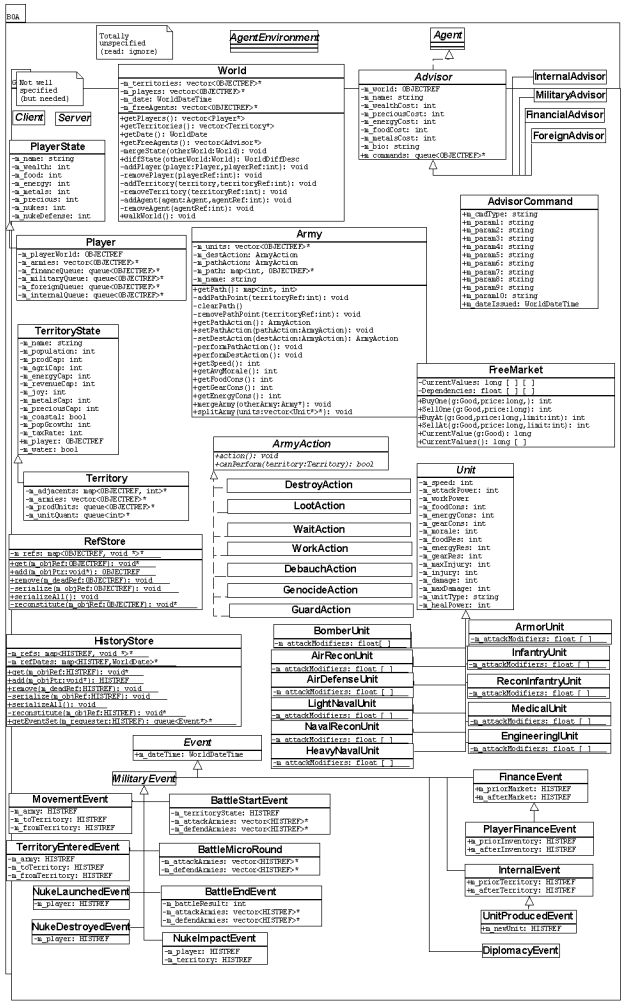
Is Software development different?



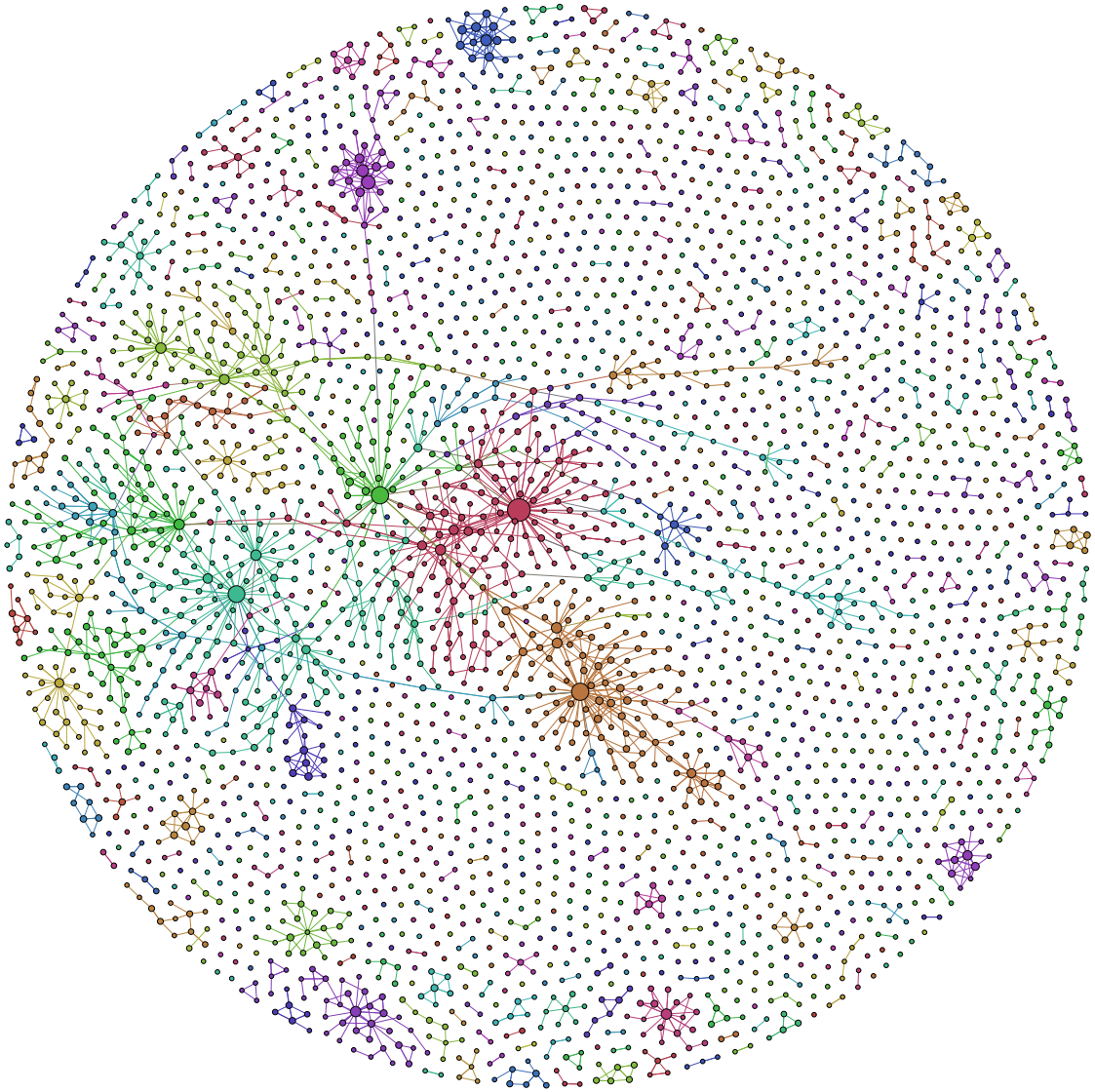


Is Software development different?

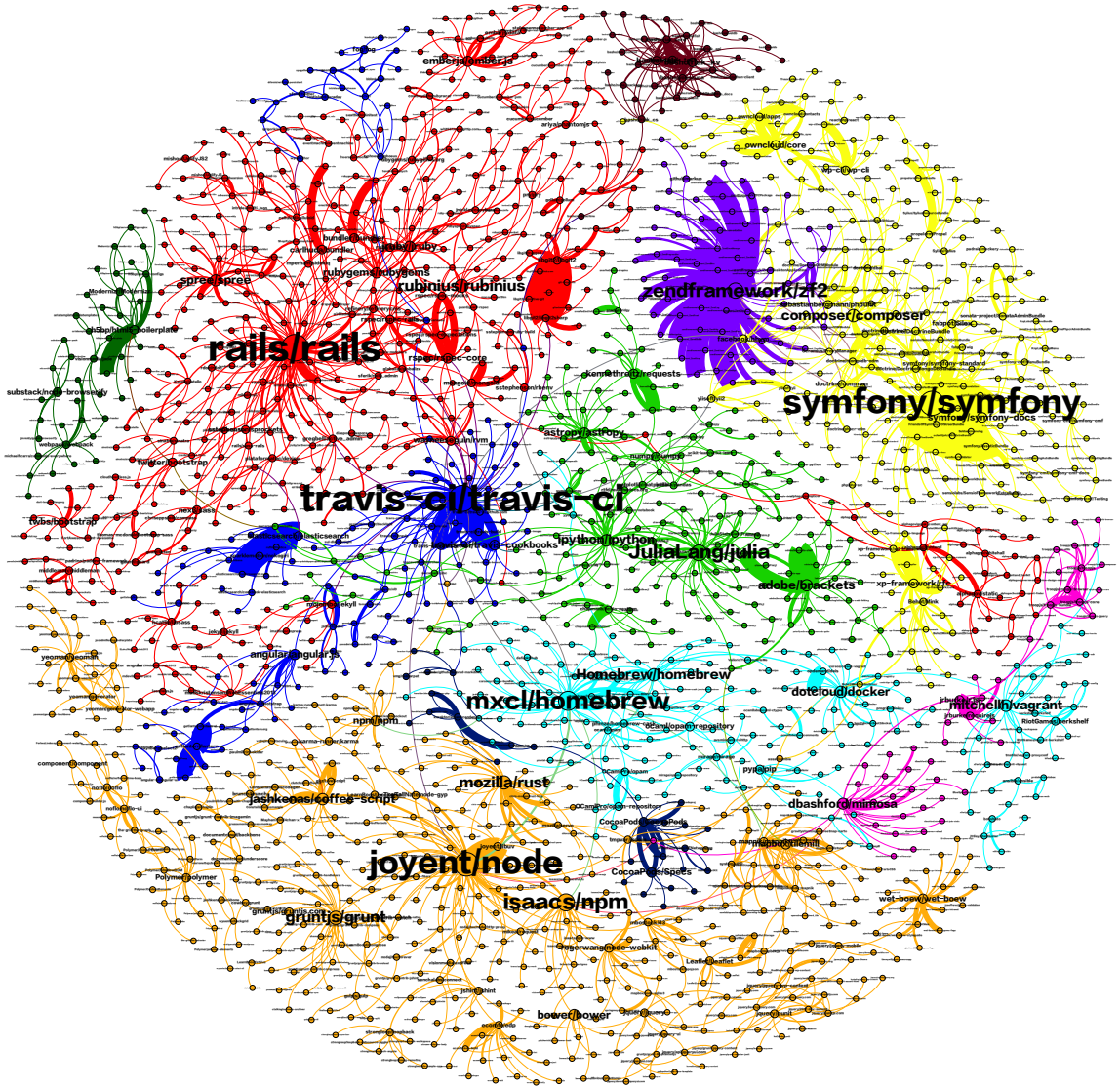




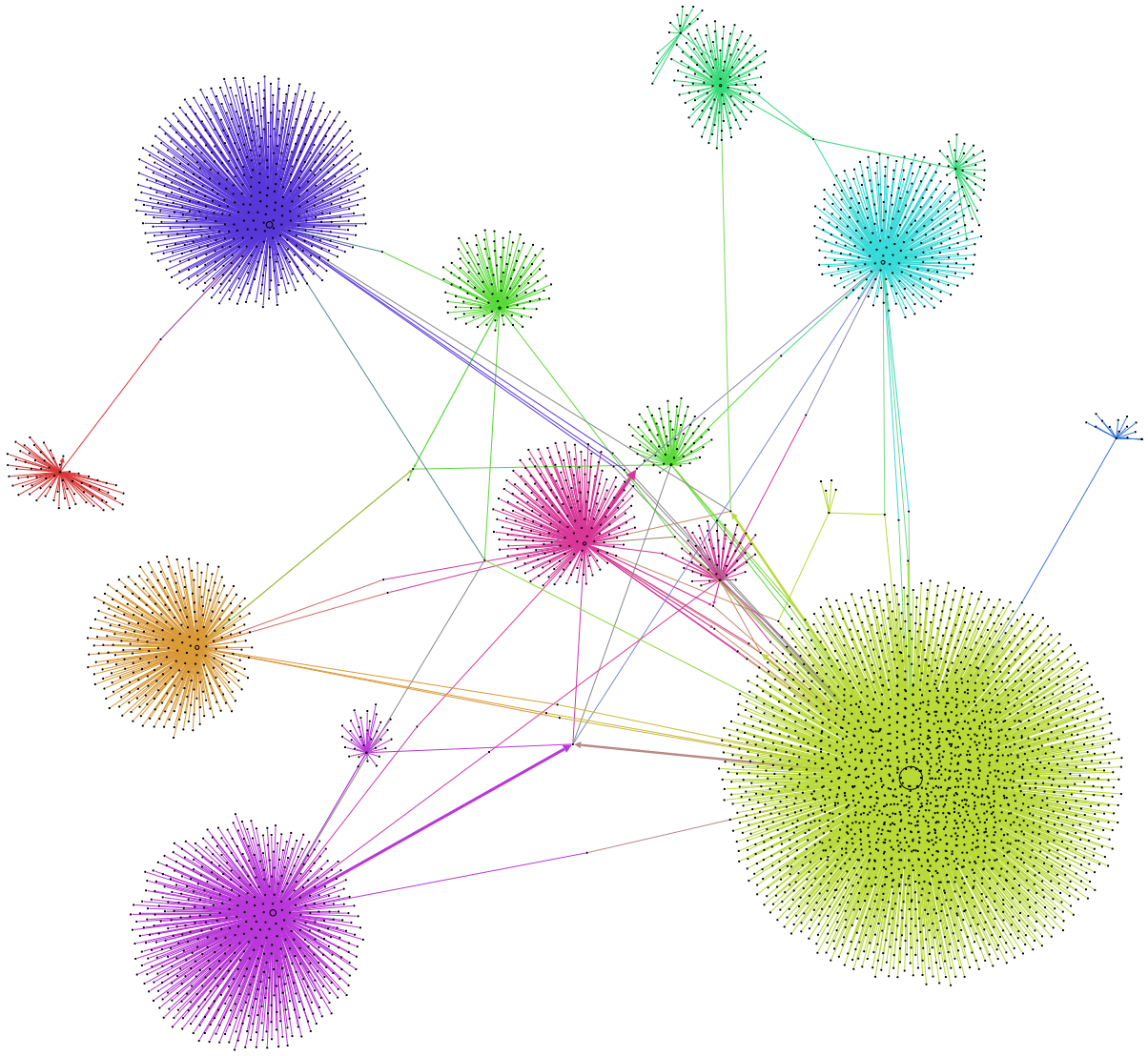
A (not so interesting) view of Github



Github dependencies: delving in



Social dependencies in Github



University of Victoria
Department of Computer Science

The research I do

I lead the Software Engineering Research Laboratory (SEGAL) on the 5th floor

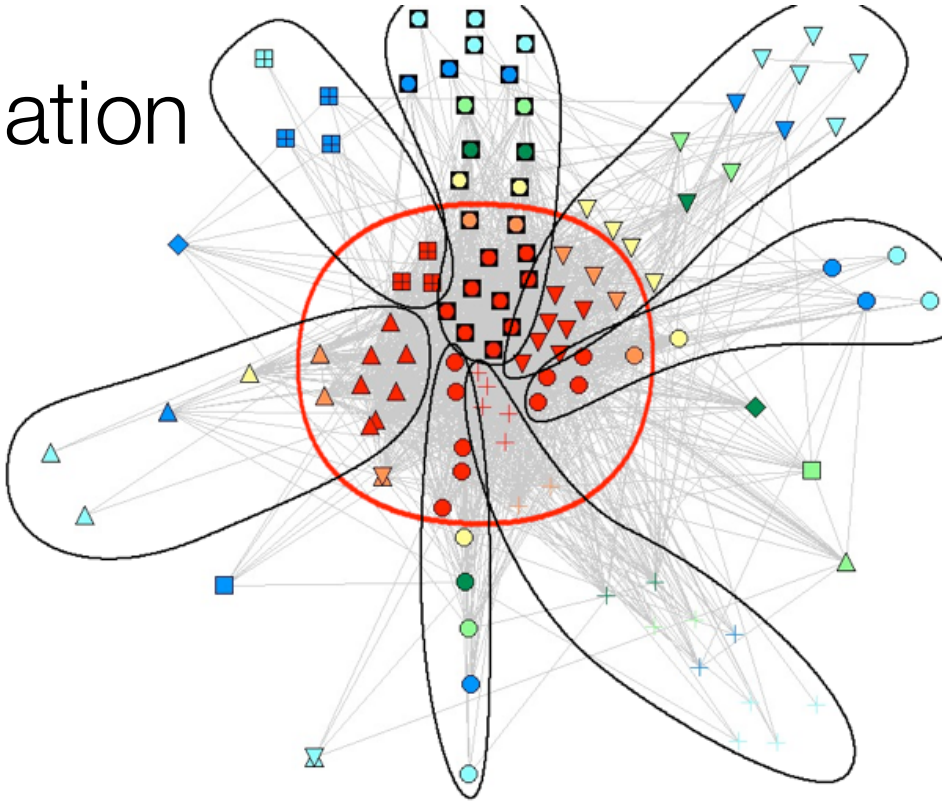
The research I do

- Socio and technical relationships in Software Development
- Studied extensively corporate and open source development projects
- Studied how software is developed in large, geographically distributed teams

Coordination in large IBM teams

We learned that

the quality of communication/coordination is a major influence on project success



Machine learning on large repositories

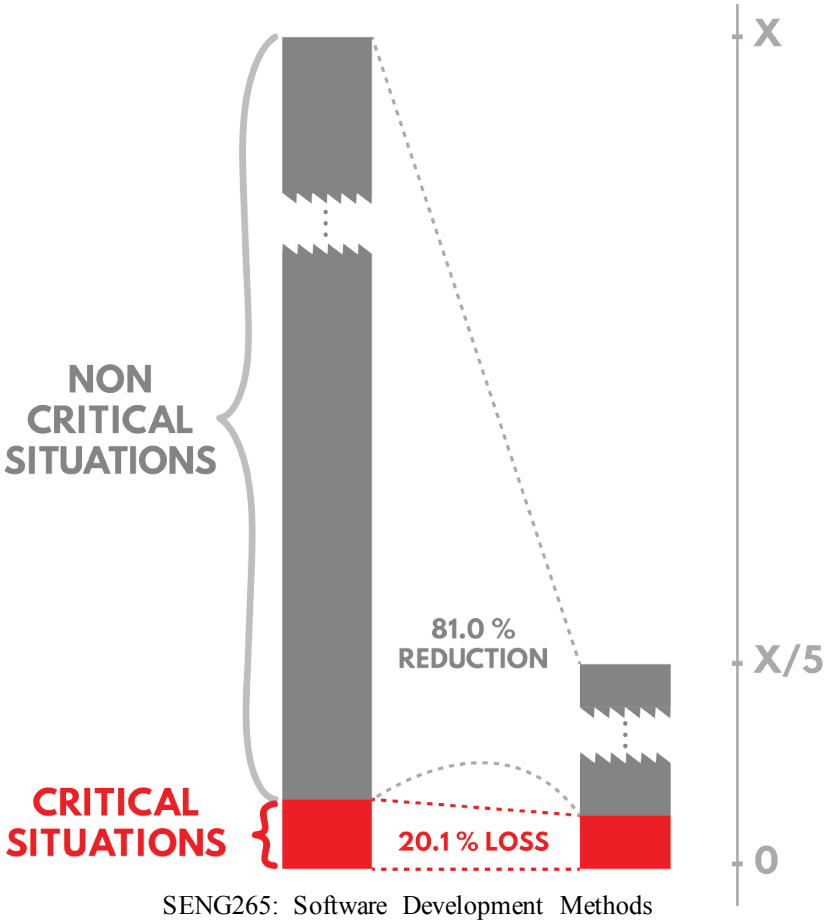
How easily can software adapt at runtime?

Machine learning on software data

Recent Best Paper award at RE conf in Lisbon

Predicted Customer issues

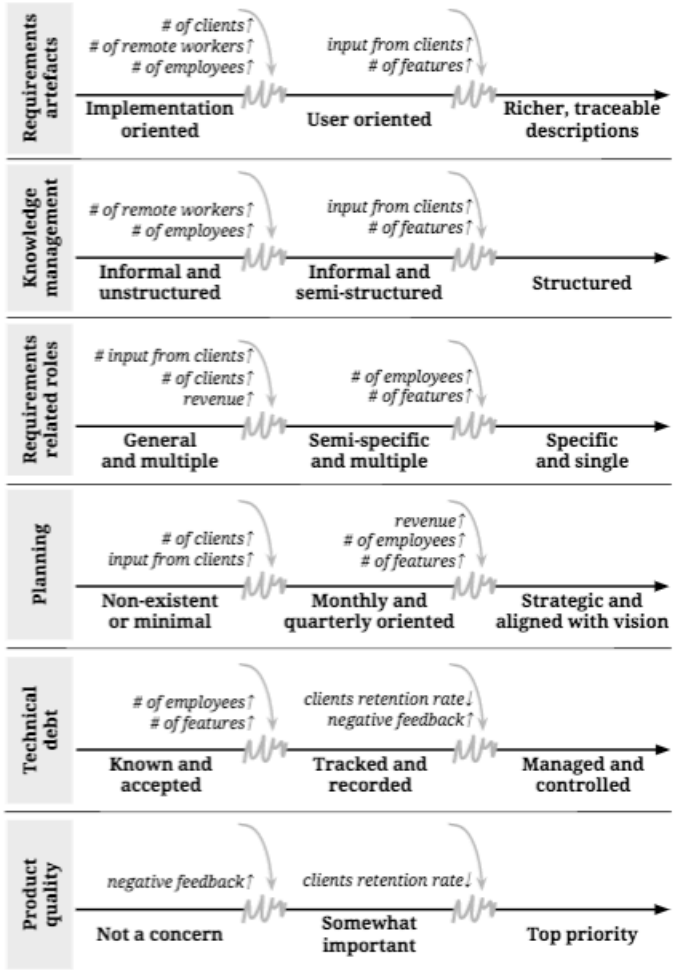
Escalations at IBM



On agile development practices

Recent study of **16 startup companies** in Victoria and other countries

Tracked the **evolution** of their development practices



The courses I teach

- OO design, Systems Analysis, Requirements Engineering, Computer Supported Cooperative Work

- **GLOBAL SOFTWARE DEVELOPMENT (see companion slides for details)**

In SENG265

- Learning environment through great communication and relationships
- A wide spectrum of knowledge and experience
- A great TA team this term

SENG265: course material in connex

- Lecture slides topic_DRAFT.pdf then _FINAL
- Lab slides: one topic per week
- Lab slides: available on the weekend before the respective week

- See Lab Topics and Schedule in Connex

SENG265: labs, TAs and office hours

Teaching Assistants:

Amanda Dash

David Johnson

Kenneth Walker

Harpreet Singh

Dana's office hours:

WT 9:00-10:00 ECS558

Lab hours	MONDAY	WED	THURSDAY	FRIDAY
10:30-11:50		LAB		LAB
11:30-12:50	LAB			
12:00-1:20		LAB		LAB
1:00-2:20	LAB		**Office hours** In the weeks before Assignments are due: **Oct 12, oct 26, nov 16**	
1:30-2:50		LAB		
2:20-3:20	**Office Hours** (Dave Johnson)			
3:00-4:30		**Office Hours** (Amanda Dash)		**Office Hours** in the weeks before Assignments are due: **Oct 13, oct 27, nov 17**
4:30-5:50	LAB	LAB		

SENG265: My expectations

- Be curious, enthusiastic, independent, perseverant, good team player; read, code, read, code, ask for help!

- When in need, communicate with (in this order):
  - Your colleagues
  - TA (in the lab, office hours, lab slack channel)
  - Assistance Center
  - Your (busy) prof (do not expect response the same day)

Effective communication - *email*

- Explain the problem, context of assignment/exercise and your difficulty/errors
- Include ”seng265” in subject line
- Be respectful and professional
- Example of a good email message asking for help (pdf file in Lab Slides->Intro folder)

Effective communication - *slack*

- Each lab will have a *slack (Mattermost) channel*
- One channel for entire class for lecture material clarifications
- Q&A style, our own StackOverflow (use the power of crowds)
- Do not post assignment code
- But could give general example suggestions to questions
- Use the Reply option to localize answers
- We will monitor professionalism and effectiveness (will turn it off if it gets out of hand)

A quick look at the schedule

- Please bring your laptop next time – we'll try some online questions
- One more lecture on Unix, then Git!