INF219 – Software Environments

First class
Introduction to the course

Marco Aurélio Gerosa

University of California, Irvine

Spring/2014

The instructor

Ph.D. in Computer Science (2006)

Bachelor in Computer Engineering (1999)

Associate Professor at the University of São Paulo, Brazil

Visiting Professor at UCI

Research area: Intersection of Software Engineering and CSCW

Web site: http://www.ime.usp.br/~gerosa

Email: mgerosa@uci.edu / gerosa@ime.usp.br

Office: DBH 5228 (by appointment)

Where I came from

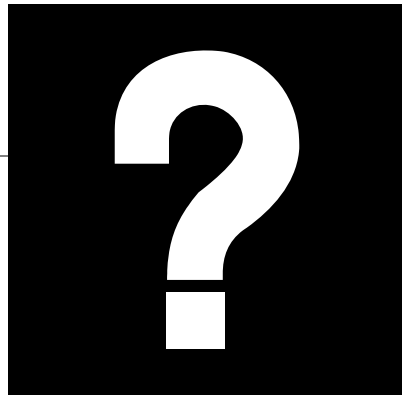
University of São Paulo

- Latin America largest university
- Total area: 76 million m^2 (800 million ft^2)
- 92000 students
- 5800 faculty
- 2400 doctorate degrees per year
- 25% of Brazilian scientific production

Computer Science Department

São Paulo

http://www5.usp.br/en/usp-em-numeros/



Who are you?

- Your name
- A little bit about yourself
- Your advisor
- Your research interests
- Experience with software environments

About the course

Logistics

Room: ICS 180

Lectures: Mondays and Wednesdays, 2pm - 3:20pm

Number of students: 17 (so far…)

Website: http://www.ime.usp.br/~gerosa/classes/uci/inf219

Additional resources: http://eee.uci.edu

How can we interact online?

- UCI EEE
- Google Groups
- Facebook

Before the content,
let's discuss **the form**

- How to memorize, understand, and be able to apply, analyze, synthesize, and evaluate the knowledge covered by the course?
- And how to develop academic skills at the same time?

Practice! Do it by yourself!

Learning...

"I hear and I forget.

I see and I remember.

I do and I understand."

(Confucius)

Learning...

"I hear and I forget.

I see and I remember.

I do and I understand."

(Confucius)

This class and the other **lectures**

Readings and the examples presented

Papers **discussion** and presentation, student **seminar**, term **project**

Attitude I: Just be present at the lectures

Attitude II: Pay close attention and be quiet

http://pidsfriendz.wikispaces.com

Attitude III: What I really want

http://chrislindholm.wordpress.com/

Back to the course

- Lectures: their content is doomed to be forgotten, but hopefully it will stay in short memory enough time to support and guide the subsequent discussion
=> Be attentive

- Readings: prepare yourself to the class to come
=> **study and synthetize knowledge**

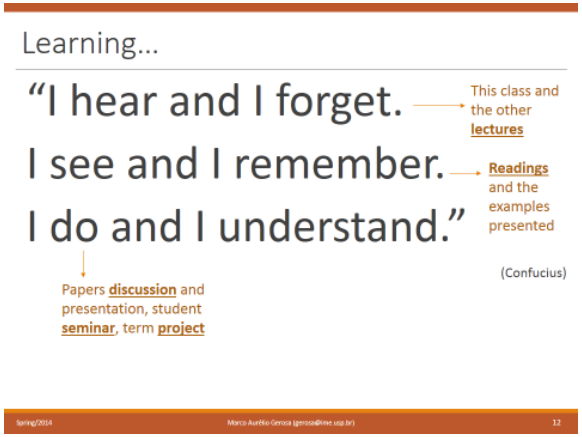
- Discussions: formalize your thoughts and share your ideas with us
=> **participate**

- Seminar: understand at least one of the course's topics in depth
=> **go in depth and show quality in what you do**

- Term project: apply the concepts in a piece of research
=> **practice and learn**

Role of the instructor => Mediator of these processes

Remember:



1. Reading and reporting

- Read 1 to 3 papers before most of the classes (most typically **1 paper**)
- Read **critically**
- Prepare a **set of slides** summarizing the paper
  - focus on the motivation, applicability, and results
  - conclude with your opinion about the paper and with a critical view on its positive and negative aspects
  - state questions and problems to be further discussed in the classroom
- In the beginning of the class, one student will be randomly selected to give an **8-10 min presentation** about the paper using his/her own slides
  - So, be prepared!!!
- Submit in advance your slides at EEE Dropbox

2. Discussions

Remember: **I want to hear you!**

2. Discussions

Remember: **I want to hear you!**

No exceptions!

Prepare yourself to the discussion as you would prepare yourself to a meeting at your job

3. Student Seminars

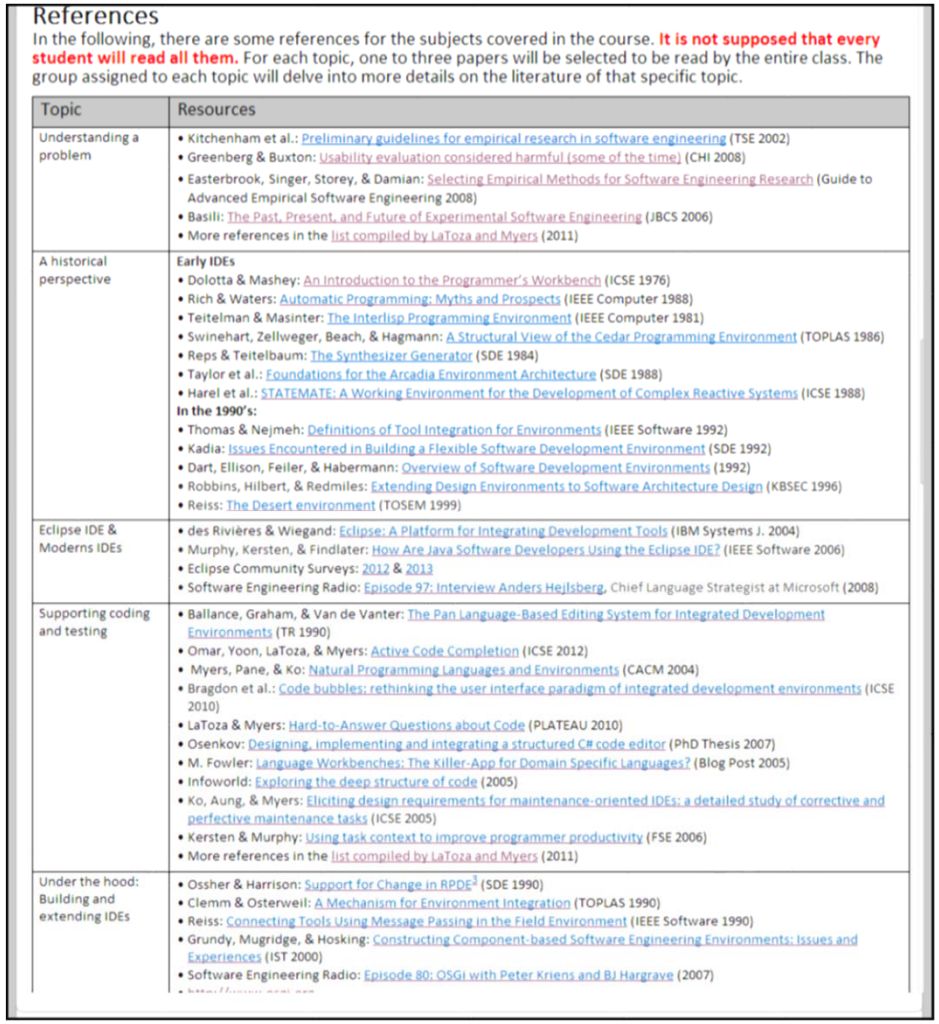
In group, study and present one topic in depth

- **Read** the literature we selected
- **Search** the literature and find additional important studies we missed
- Map the **main concepts** about the topic
- Identify **important and interesting works** published in the literature
- Investigate the **state-of-the-practice**, actual examples, and tools used in **software industry**
- Identify **one or two key articles** for the class to read
  - Let us know at least 1 week before your scheduled presentation date
  - One random student will do the 10 min presentation about the paper
- Compile a **categorized bibliography** about the topic
- Prepare a **30 minute overview** of the area using slides
- Prepare **questions and challenges** to motivate the discussion in class

We're looking forward to learn about the area **from your perspective**!

References

In the website, there is an **initial list of relevant references**. Let's improve it!



References

In the following, there are some references for the subjects covered in the course. **It is not supposed that every student will read all them.** For each topic, one to three papers will be selected to be read by the entire class. The group assigned to each topic will delve into more details on the literature of that specific topic.

Topic	Resources
Understanding a problem	• Kitchenham et al.: Preliminary guidelines for empirical research in software engineering (TSE 2002) • Greenberg & Buxton: Usability evaluation considered harmful (some of the time) (CHI 2008) • Easterbrook, Singer, Storey, & Damian: Selecting Empirical Methods for Software Engineering Research (Guide to Advanced Empirical Software Engineering 2008) • Basili: The Past, Present, and Future of Experimental Software Engineering (JBCS 2006) • More references in the list compiled by LaToza and Myers (2011)
A historical perspective	**Early IDEs** • Dolotta & Mashey: An Introduction to the Programmer's Workbench (ICSE 1976) • Rich & Waters: Automatic Programming: Myths and Prospects (IEEE Computer 1988) • Teitelman & Masinter: The Interlisp Programming Environment (IEEE Computer 1981) • Swinehart, Zellweger, Beach, & Hagmann: A Structural View of the Cedar Programming Environment (TOPLAS 1986) • Reps & Teitelbaum: The Synthesizer Generator (SDE 1984) • Taylor et al.: Foundations for the Arcadia Environment Architecture (SDE 1988) • Harel et al.: STATEMATE: A Working Environment for the Development of Complex Reactive Systems (ICSE 1988) **In the 1990's:** • Thomas & Nejmeh: Definitions of Tool Integration for Environments (IEEE Software 1992) • Kadia: Issues Encountered in Building a Flexible Software Development Environment (SDE 1992) • Dart, Ellison, Feiler, & Habermann: Overview of Software Development Environments (1992) • Robbins, Hilbert, & Redmiles: Extending Design Environments to Software Architecture Design (KBSEC 1996) • Reiss: The Desert environment (TOSEM 1999)
Eclipse IDE & Moderns IDEs	• des Rivières & Wiegand: Eclipse: A Platform for Integrating Development Tools (IBM Systems J. 2004) • Murphy, Kersten, & Findlater: How Are Java Software Developers Using the Eclipse IDE? (IEEE Software 2006) • Eclipse Community Surveys: 2012 & 2013 • Software Engineering Radio: Episode 97: Interview Anders Hejlsberg, Chief Language Strategist at Microsoft (2008)
Supporting coding and testing	• Ballance, Graham, & Van de Vanter: The Pan Language-Based Editing System for Integrated Development Environments (TR 1990) • Omar, Yoon, LaToza, & Myers: Active Code Completion (ICSE 2012) • Myers, Pane, & Ko: Natural Programming Languages and Environments (CACM 2004) • Bragdon et al.: Code bubbles: rethinking the user interface paradigm of integrated development environments (ICSE 2010) • LaToza & Myers: Hard-to-Answer Questions about Code (PLATEAU 2010) • Osenkov: Designing, implementing and integrating a structured C# code editor (PhD Thesis 2007) • M. Fowler: Language Workbenches: The Killer-App for Domain Specific Languages? (Blog Post 2005) • Infoworld: Exploring the deep structure of code (2005) • Ko, Aung, & Myers: Eliciting design requirements for maintenance-oriented IDEs: a detailed study of corrective and perfective maintenance tasks (ICSE 2005) • Kersten & Murphy: Using task context to improve programmer productivity (FSE 2006) • More references in the list compiled by LaToza and Myers (2011)
Under the hood: Building and extending IDEs	• Ossher & Harrison: Support for Change in RPDE^3 (SDE 1990) • Clemm & Osterweil: A Mechanism for Environment Integration (TOPLAS 1990) • Reiss: Connecting Tools Using Message Passing in the Field Environment (IEEE Software 1990) • Grundy, Mugridge, & Hosking: Constructing Component-based Software Engineering Environments: Issues and Experiences (IST 2000) • Software Engineering Radio: Episode 80: OSGi with Peter Kriens and BJ Hargrave (2007)

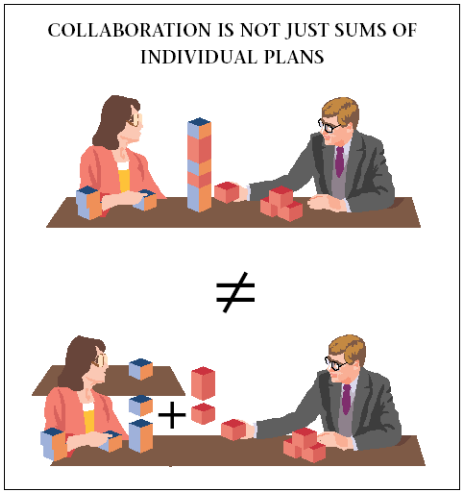
Groups

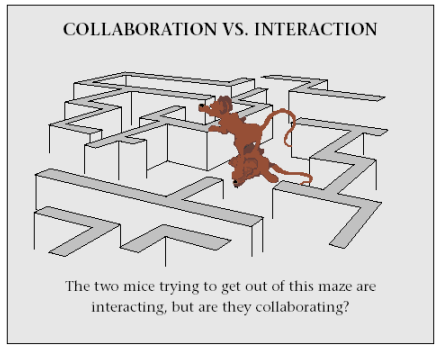
Groups of 2 (or 3 depending on the number of enrolled students) – preferably mixing Ph.D. and MSc students

Define your group, post it in the shared document https://docs.google.com/document/d/1KwRp4Gs5zO0PxBMXXDnb6UeKj4JNx-Tp2_llWaBeH6A/edit?usp=sharing and send me an email just in case (mgerosa@uci.edu)

Topics are chosen in a first-to-come-first-to-serve basis => **act quickly!**

Real collaboration is expected!





Grosz. Collaborative Systems. *AI Magazine*, 17(2): 67-85. Summer 1996

4. Term project

Objective: to conduct a small research project and write a paper about it

What is a good research project involving software environments?

Remember: We want to support software developers

Software developers
are **human beings**,
not machines

What is a good research in the area?

Technical aspects are not enough!

A good research needs to consider the **human and social aspects** of software development.

So, a **good research** comprises [Myers, 2011]:

- Contextual inquiry or lab study to discover an interesting issue that has not previously been known
- Survey to validate that is actually widespread
- A model that represents and generalizes what is happening
- Tool developed that embodies interesting technical contributions and addresses the problems
- Lab study of the tool, compared to the way it is done now, that shows significant improvements

Brad Myers (2011) http://www.cs.cmu.edu/~bam/uicourse/2011hasd/lecture01.intro.pptx

So, what do you need to do?

- **Choose a topic**
- **Conduct an empirical evaluation** with humans to understand the problem to be solved (more details next class)
- **Survey the literature** (related work) **and software repositories** to find evidences of the relevance of the problem
- **Conceive a solution**
- **Implement it**
- **Evaluate it**
- **Report** (write an academic paper)

The topic

Substantial latitude is allowed, but it has to be related to software environments and must consider human/social aspects of software development

Tip: consider something related to your seminar or to your graduation theme

Small changes are allowed along the quarter (the project is iterative and incremental)

The research proposal must be approved by the instructor

Proposal: motivation, research question, objective, method (more details about it on the second class) – 1 or 2 pages

Due date: April, 9th

Feedback on the project (I)

Some classes will be used for the projects follow up

All the students are expected to give constructive feedback during classes and/or in the online discussion

April 16th: Topics and empirical evaluations

April 30th: Solution proposed

May, 21th: Prototype

Jun, 11th: Final presentation

Feedback on the project (II)

The academic paper will be submitted via Easy Chair, simulating a **scientific conference**

You will be part of the **program committee** of the conference

You are going to receive papers from the other groups **to review**

The reviews will be considered as part of the reviewer's grade, but **will not affect negatively the grade of the authors** of the papers.

- Therefore, you can be critical (but always in a constructive way) without prejudicing the colleagues

The reviews will be anonymous

Grading

What will be considered:

- Presence and participation in the classes and in the online discussion
- Critical opinions, questions, and challenges shared with the class on the discussions
- Feedback provided to your peers
- The demonstrated understanding of the class topics
- The quality of what you produce in the courses activities:
  - Summary slides
  - 8-10 min paper presentations
  - Seminar
  - Term project

Finally, the content

Description

Catalogue description: Study of the requirements, concepts, and architectures of comprehensive, integrated, software development and maintenance environments. Major topics include process support, object management, communication, interoperability, measurement, analysis, and user interfaces in the environment context.

This edition's scope: This edition highlights **human and social aspects** of software engineering, as well as **information mining from software repositories**. We are going to discuss how software environments can be improved considering these elements.

Key aspects:

- Understanding software development problems

- Conceiving tools to help developers

- Evaluating these tools

What are we going to study?

A historical perspective

How to improve software environments

Next steps and future of the area

Focus on research, but without forgetting the state-of-the-practice

Topics

Understanding a problem

A historical perspective

Eclipse IDE & Moderns IDEs

Supporting coding and testing

Under the hood: Building and extending IDEs

Supporting software analysis and design

Version Control Systems & Configuration Management

Mining software repositories

Software analytics

Software visualization

Integrating IDEs and Social Media

Distributed Software Environments

IDEs @ the Cloud

Introduction to Software Environments

What is a software environment?

Write down what you consider to be a software environment

What is a software environment?

Refers to the collection of hardware and software tools a system developer uses to build software systems.

Dart et al. (1992)

The set of software tools collected together (sometimes using a common database or user interface) for use by a software developer, or team of developers, when developing software.

Oxford Dictionary of Computing

An environment is a collection of CASE tools and workbenches that supports the software process.

http://en.wikipedia.org/wiki/Computer-aided_software_engineering

Many related terms

CASE tool (Computer-Assisted Software Engineering) and ICASE (Integrated CASE)

- A marketing term, used to describe the use of software tools to support software engineering

IPSE (Integrated Project Support Environment)

- A software system that provides support for the full life cycle of software development and also the project control and management aspects of a software-intensive project.

SEE (Software Engineering Environment) and ISEE (Integrated Software-Engineering Environment)

- Concern for the entire software life cycle (rather than just program development) and offer support for project management (rather than just technical activities)

SDE (Software Development Environment)

- The set of software tools collected together (sometimes using a common database or user interface) for use by a software developer, or team of developers, when developing software

PSE (Programming Support Environment) or Programming Environment

- A software system that provides support for the programming aspects of software development, repair, and enhancement. A typical system contains a central database and a set of software tools. A programming support environment might be considered as a more technologically advanced form of PDS.

Oxford Dictionary of computing, 6th ed.
http://www.oxfordreference.com/view/10.1093/acref/9780199234004.001.0001/acref-9780199234004

Many related terms

PDS (Program Development System)

- A software system that provides support to the program development phase of a software project.

Workbench

- Another name for software development environment

Toolbox and Toolkits

- A set of software tools, probably from several vendors, not necessarily as closely related or providing as full coverage of the software life cycle as a toolkit. The set of tools in a toolkit is usually from a single vendor.

SDK (Software Development Kit)

- A collection of the software tools, code libraries, documentation, etc., necessary to develop a specific type of software, commonly provided as a single installable package.

Software tool

- A program that is employed in the development, repair, or enhancement of other programs

Oxford Dictionary of computing, 6th ed.
http://www.oxfordreference.com/view/10.1093/acref/9780199234004.001.0001/acref-9780199234004

Why automated tools in software environments?

CASE tools vs. notepad, vi, etc.

Some benefits

Increase productivity

Increase quality

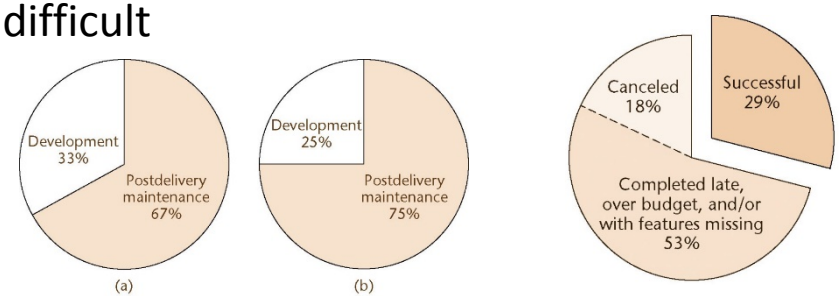
Automate software processes

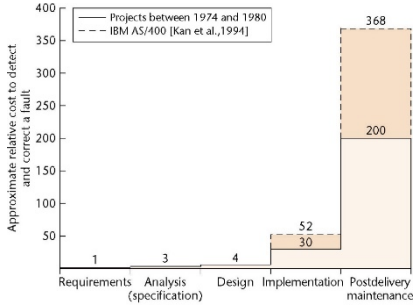
Reduce costs

Foster collaboration

Support management

Yes, we know that software engineering is difficult





[Schach, 2009]

And involves a lot of stakeholders

If we have benefits, we have disadvantages too. What are they?

Possible problems

Learning curve

Dependency on third parties vendors

Cost of the solution (Total Cost of Ownership)

What is an Integrated Development Environment (IDE)?

A definition

Tool integration is about the extent to which tools agree. The subject of these agreements may include data format, user-interface conventions, use of common functions, or other aspects of tool construction.

Thomas & Nejmeh: Definitions of Tool Integration for Environments (IEEE Software 1992)

An IDE normally consists of a source code editor, build automation tools, and a debugger.

The boundary between an integrated development environment and other parts of the broader software development environment is sometimes blur.

Integrated development environments are designed to maximize programmer productivity by providing tight-knit components with similar user interfaces.

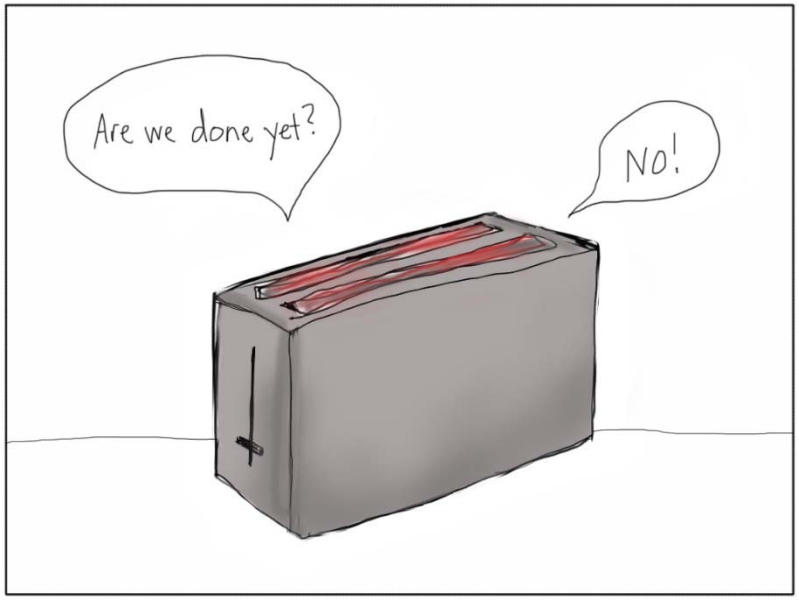
One aim of the IDE is to reduce the configuration necessary to put together multiple development utilities

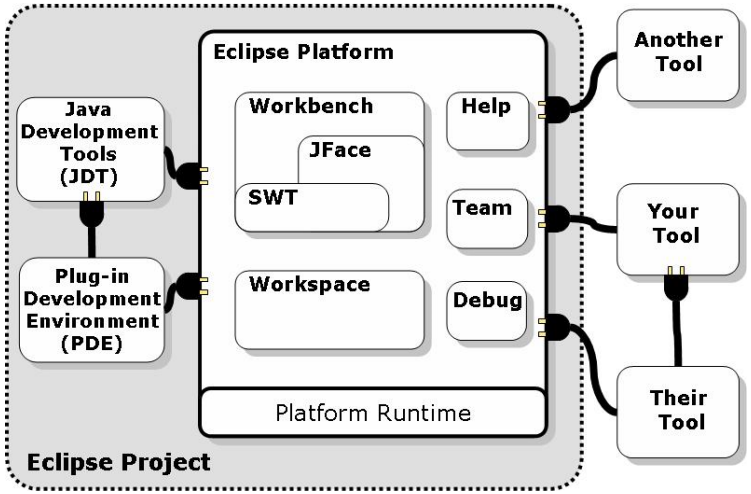
http://en.wikipedia.org/wiki/Integrated_development_environment

Extending IDEs

Component-based software development

Plugins





Different software environments

- Focus on a programing language / technology
- Focus on a process / method
- Focus on a domain
- Focus on an organization

Some more definitions

Program
- A set of statements that can be submitted as a unit to some computer system and used to direct the behavior of that system

Programming
- The process of transforming a mental plan of desired actions for a computer into a representation that can be understood by the computer

Development
- All programmer activities involved in software (except use)
- Includes design, programming (coding), testing, documentation, etc.

Software Engineering
- Development + Requirements Analysis + processes

End-User Development
- End users doing development activities themselves

Visual Programming
- Programming using graphics
- 2D layout is significant (beyond indentation)

Interface Builders
- Draw the layout of static parts of a user interface, and code is generated to create that layout at run-time

Brad Myers (2011) http://www.cs.cmu.edu/~bam/uicourse/2011hasd/lecture01.intro.pptx

Other related terms

Library

- A collection of programs and packages that are made available for common use within some **environment**

Framework

- A template for the development of software applications or components

Application Programming Interface (API)

- A body of code providing reusable functionality that is intended to be used without looking inside at the internal implementation

Domain-Specific Languages (DSL)

- Programming languages designed for a particular audience

Brad Myers (2011) http://www.cs.cmu.edu/~bam/uicourse/2011hasd/lecture01.intro.pptx

Oxford Dictionary of computing, 6th ed.
http://www.oxfordreference.com/view/10.1093/acref/9780199234004.001.0001/acref-9780199234004

Some definitions

Human Aspects of Software Development

- All the parts of development that affect the developer
- Not the parts that affect the end user

Mining Software Repositories

- Analysis of data available in software repositories to uncover interesting and actionable information about software systems and projects

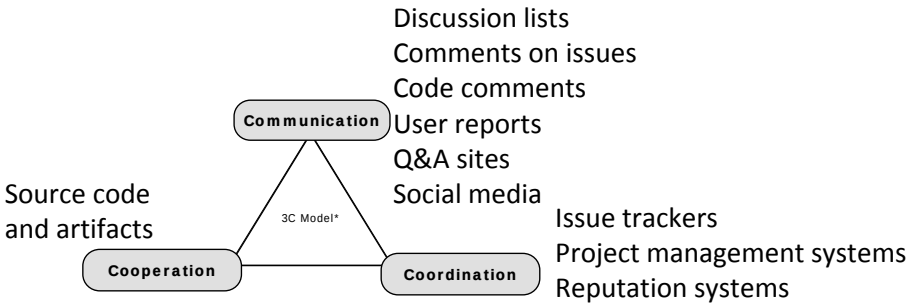
http://2014.msrconf.org/

Brad Myers (2011) http://www.cs.cmu.edu/~bam/uicourse/2011hasd/lecture01.intro.pptx

Collaboration tools and social media can be part of a software environment?

Remembering the definition: Software Environment refers to the collection of hardware and software tools a system developer uses to build software systems.

Mining software repositories



Mining

Information about a project

Information about an ecosystem

Information about Software Engineering

Applications

Decision making

Software understanding

Support maintenance

Empirical validation of ideas & techniques

Repositories of repositories

11.3 millions repositories
5.4 millions users
In 2013:

- 3 millions new users
- 152 millions pushes
- 25 millions comments
- 14 millions issue
- 7 millions pull requests

https://github.com/about/press
http://octoverse.github.com/

250K projects

93K projects
1 million users

661K projects
29 billions of lines of codes
3 millions users

http://www.ohloh.net/

30K projects

https://launchpad.net

36K projects

http://en.wikipedia.org/wiki/CodePlex

33K projects

324K projects
3.4 millions developers

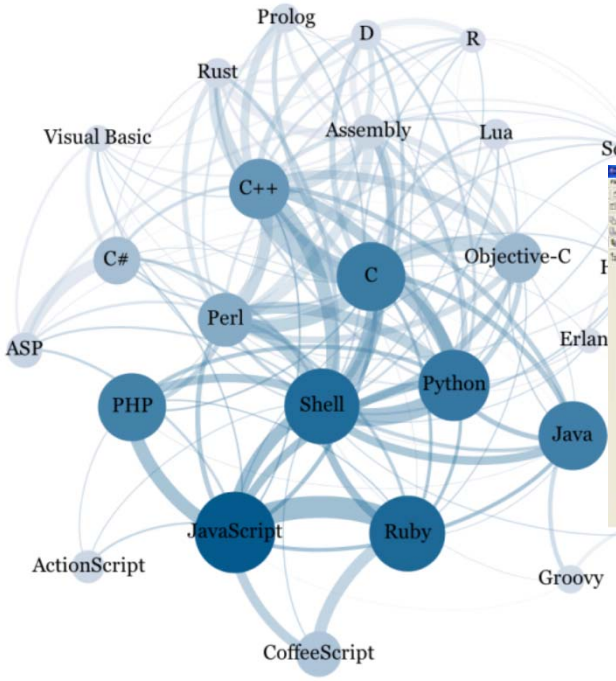
http://sourceforge.net/apps/trac/sourceforge/wiki/What%20is%20SourceForge.net

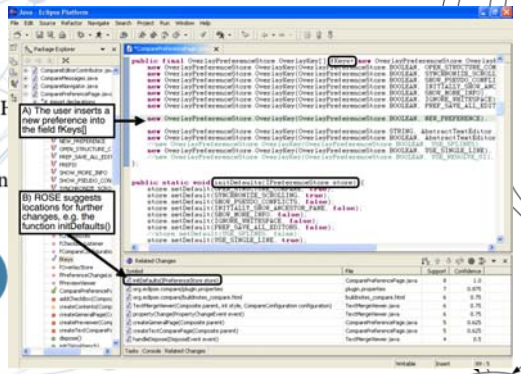
200 projects

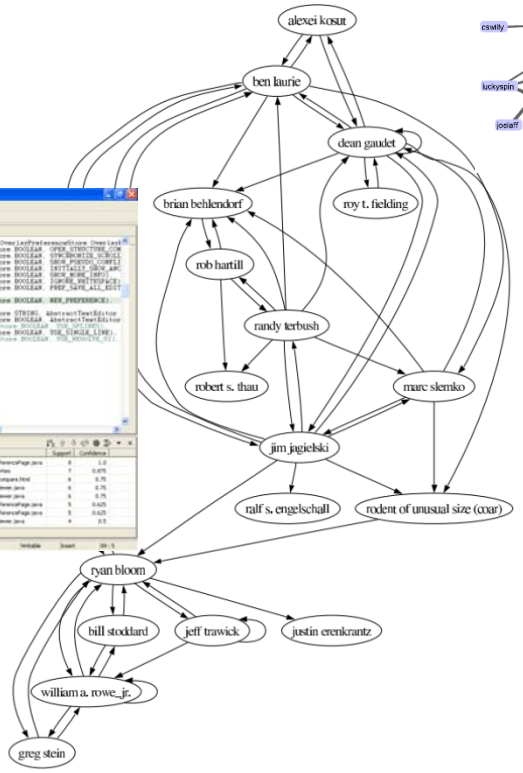
http://projects.apache.org/indexes/alpha.html

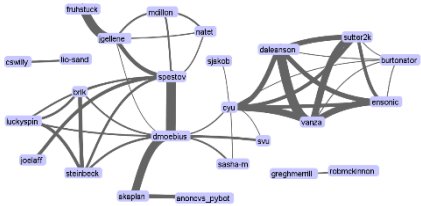
http://en.wikipedia.org/wiki/Comparison_of_open_source_software_hosting_facilities

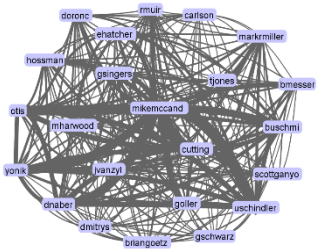
Examples











A **Ruby** programmer is ***very likely to know*** **JavaScript**, while a **Perl** programmer is not.

Java is a popular language, but stands primarily alone.

https://github.com/mjwillson/ProgLangVisualise
http://www.igvita.com/slides/2012/bigquery-github-strata.pdf

Zimmermann, Weissgerber, Diehl, and Zeller. Mining Version Histories to Guide Software Changes. IEEE Trans. Software Eng. 31, 6 (June 2005)

Christian Bird, Alex Gourley, Prem Devanbu, Michael Gertz, and Anand Swaminathan. 2006. Mining email social networks. MSR 2006

Santana, F. et a. "XFlow: An Extensible Tool for Empirical Analysis of Software Systems Evolution". ESELAW 2011

Conclusion

The other side of the coin

"One of the things I've found kind of depressing about the craft of computer programming is that you see all these fancy development environments out there, and they tend to be targeted at solving certain specific kinds of problems, like building user interfaces. But **if you go around and look at the high-end developers, the people who actually have to implement algorithms, if you're using one of these high-end IDEs the IDEs generally don't help you much at all, because they drop you into this simple text editor.** The number one software development environment for high-end developers these days is still essentially EMACS. At their heart, these tools are 20 years old, and there hasn't been much in the way of dramatic change. People have made all kinds of stabs at graphical programming environments and that, and they've tended to be failures for one reason or another."

Why is it that people still use ASCII text for programs? It just feels like there's so much territory out there that's beyond the bounds of ASCII text that's just line after line, roughly 80 characters wide, mostly 7-bit ASCII, something that you could type in on a teletype."

http://www.gotw.ca/publications/c_family_interview.htm

James Gosling

http://en.wikipedia.org/wiki/James_Gosling

Where to look for research in this area?

Scientific venues

- ICSE, CHI, FSE, UIST, CSCW, TSE, TOSEM, ICSM, ICPC, JVLC, Journal of Systems & Software, etc.

Search engines

- http://scholar.google.com
- http://dblp.uni-trier.de/db
- http://www.scopus.com
- http://ieeexplore.ieee.org
- http://dl.acm.org
- http://citeseer.ist.psu.edu

Tip: Use a tool to manage your references, such as http://www.zotero.org, http://www.mendeley.com, or http://jabref.sourceforge.net

To Do

- Read the guidelines of the course at the website **http://www.ime.usp.br/~gerosa/classes/uci/inf219** (these slides will also be there)
  - Although I will most probably report the major updates about the course in the discussion list, you can monitor this page with an online service such as http://www.changedetection.com/ or http://www.watchthatpage.com/
- Define the groups and the topic of your seminar
- Read the paper assigned for the 2nd class (check the webpage)
- Produce the slides-based summary of the paper

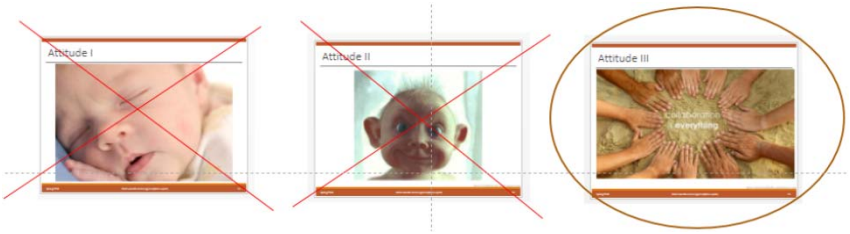
"Wherever you go, go with all your heart." (Confucius)

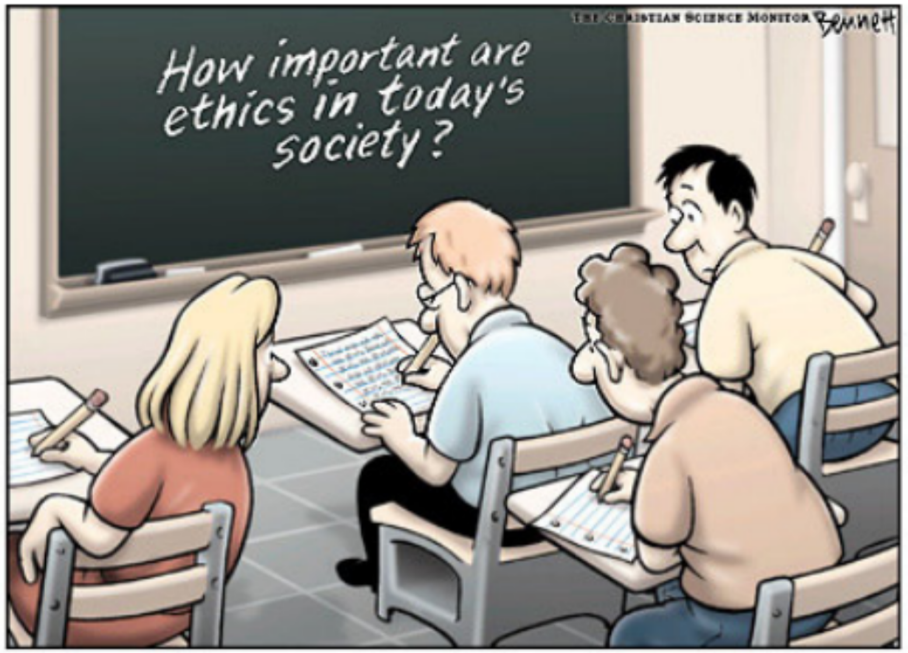
Fair Play!

Do not cheat

Do not copy other works without proper citation

Do not overload the other(s) member(s) of the group





http://www.claybennett.com/pages/ethics.html

See also:

http://honesty.uci.edu/students.html

Summary

- Who we are
- The dynamics of the course
  - Attitude
  - Assignments
  - Topics
- What is a software environment
- Terminology
- Tool support for Software Engineering
- IDEs
- Mining software repositories
- Where to find research on the area
- Fair play

Thank you!

Marco Gerosa
Web site: http://www.ime.usp.br/~gerosa
Email: mgerosa@uci.edu / gerosa@ime.usp.br
Office: DBH 5228 (by appointment)

Acknowledgments

This course is based on other courses from André van der Hoek, Susan Elliott Sim, Myers and LaToza, and Leonardo Murta. I also would like to thank the feedback received from David Redmiles, Yi Wang, Andre van der Hoek, Thomas LaToza, Gustavo Oliva, Igor Steinmacher, and Igor Wiese.