

Practices of Global Software Development

SE 371

Fall 2014

instructor: Ed Keenan
email: ekeen2@cdm.depaul.edu
office hours: Tues 3-5pm, 9-10pm after class or by email appointment
office: CDM 830
phone: (312)362-6747
website: piazza.com/depaul/fall2014/se475 (Preferred communication)
lecture: CDM 224, Wednesday, 5:45-9:00pm
Desired to Learn (D2L): d2l.depaul.edu (Grades, Viewing lectures, Announcements)
Version Control: perforce: **140.192.39.61:1666**

Description:

This course studies the challenges and solutions for developing software in a global environment. Topics covered include strategic issues related to allocation of tasks; communication issues that arise due to distance, time zone differences, infrastructure support, geographical dispersion, and lack of information communication; coordination complexity; cultural issues, technical issues related to information and artifact sharing, and architectural design, and finally knowledge management issues. Some sections of this course will include a hand-on global development project in conjunction with teams of students at other universities, while other sections will focus on a more in-depth theoretical discussion of these issues and concepts.

Prerequisites:

- Data Structures in Java or C++ (CSC 301 or CSC 383 or CSC 393)
- or instructor consent

Learning Goals:

- Software engineering concepts for team development and project management in a globally distributed manor
 - Understanding geographic, time related, cultural, economic and management issues of global software development
- Software configuration management (SCM)
 - Understanding version control usage in a large scale project, including merging, branching, release and bug tracking
- Global software project management
 - Scheduling, estimating, coordinating, and monitoring of global base projects
 - Culturally based leadership and conflict resolutions with direct and indirect reporting
- Software Team Development in a global environment
 - Coordinating and communicating with distributed developers
 - Trouble shooting and debugging with asynchronous communication
 - Understanding collaboration tools in a remote environment

Grading:

Individual Homework

- 5% - Environment Setup
- 10% - Software Configuration Management (Version Control) assignment
- 10% - Design Journals

Team Projects

- 20% - Testing Project
- 20% - Subordinate Project
- 20% - Partitioning Project
- 0% - Continuous Project (Class Group project)

Final Exam

- 15% - Final Exam & Business Plan Project

Textbooks and printed resources

Required Texts:

Additional course material will be many supplied through class notes, handouts or online links.

- ***Smartsourcing: Driving Innovation and Growth Through Outsourcing*** (Hardcover) ~ Thomas M. Koulopoulos (Author), Tom Roloff (Author) ISBN-13: 978-1593375140
- ***The Pragmatic Programmer: From Journeyman to Master*** (Paperback) ~ Andrew Hunt ISBN-13: 978-0201616224

Additional Material

- **Research material:**
 - Post-mortem and software engineering articles will be distributed in class
- Lectures, links, SDKs and other corresponding material

Software

- **Microsoft Visual Studio 2012 (ultimate edition - Recommended)**
 - [DePaul MSDNAA link Microsoft Visual Studio Ultimate 2012](#)
 - C++ and C# install (future classes)
 - Microsoft Visual Studio 2013 is not used in this class.
- Perforce - Visual Client (p4v)
 - www.perforce.com
- Download and configuration instructions will be provided in class
 - Server address: **140.192.39.61:1666**

Version Control Programming

Using Perforce, you will do a series of check-ins and check-outs, merging the code, resolving bugs, branching, versioning, moving, deleting, monitoring

- Sufficient tasks to demonstrate a thorough knowledge of the system
- Including common troubleshooting issues
- Merge issues

Final Exam + Business Plan (Week 11)

- Final Exam covering the concepts and the material of the class.
- Business plan (see below)
- Take home exam
 - Short answer and essay

Design Journals (Blogs)

Design journals are due with each project, they will contain a work journal of the design and coding issues the student have during the course of their work. Outlining the developmental issues, communication issues and solutions you used to remedy the situation.

- Expecting 2-3 journal entries per week
 - Every time you work on project or do communication
- Visio Diagrams of structures

Class Structure:

- Students are still forming their beliefs
 - Background assumptions generally follow a manufacturing model
 - Student actually reflect many common misconceptions that happen in practice
- Globally Distributed Software Development (GDSD) is superset of software engineering
 - Every challenge exists in GDSD
 - Penalties are greater in a Remote environment
- Understanding GDSD issues
 - Helpful in understand all software development issues.
 - Every mistake is amplified
- The class is authored to understand both roles of remote development.
 - Design Team
 - Organization that hires services
 - Customer – in outsourcing model
 - Remote Team
 - Organization that provides services
 - Provider – in outsourcing model
- Use Patterns:
 - Specific patterns between the design and remote teams keep repeating.
 - Independent of company changes / acquisitions
 - Large project fund changes forced new scenarios
 - Several patterns were observed
 - Independent of the type of topology, same issues are present in
 - Distributive Development
 - In-sourcing
 - Outsourcing
- Goals
 - Condense experience
 - Taught and experienced through projects
 - Lesson built on progressively more complex user patterns
- Class is made up of five hands-on projects (one per pattern)
 - Remote Testing
 - Subordinate Role
 - Partitioning
 - Continuous Development
 - Business Plan
- Traditional Class lectures
 - Presentations, books, tests
- Remote students at another institution
 - Role play GDSD between international teams

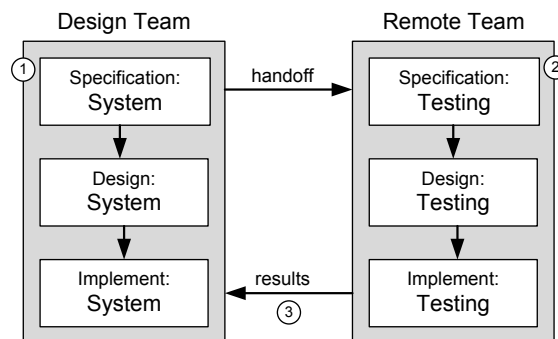
Projects:

The projects are designed to be implemented by DePaul and external university students in different groups made up of 4-5 members. Additionally student groups from DePaul will be a mixture of in-class and distance learning students. Each team will play the **Design** and **Remote** roles within each project for maximum effect.

Students will gain experience of the critical issues by implementing the project in a team environment. Through their real-world experience they form keen insights and understandings on how to run and manage similar projects in the future.

- Testing
- Subordinate
- Partitioning
- Continuous
- Business Plan

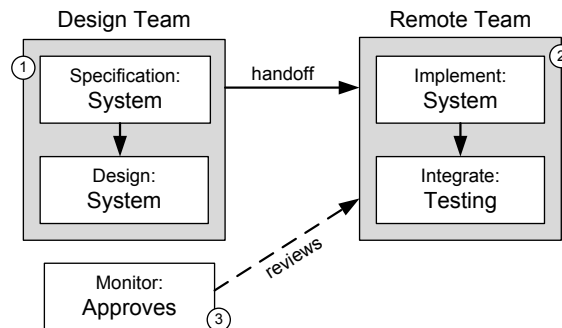
Testing Project:



- **Project:**
 - Design team delegates testing to the remote team.
 - Remote team isn't the author of the systems
- **Implement:**
 - Reverse engineer, analyze, and create tests on existing software
 - Schedule and manage the deployment of tasks
- **Learn:**
 - Centralized communication
 - Wiki, forums, collaborative tools
 - Scheduling
 - Cultural and language barriers
 - Testing
 - Types of Testing: Unit, Regression, System,
 - Testing coverage: Depth vs Breath
 - Static analysis and automatic tools
 - Working legacy code
 - Reverse engineering
 - Refactoring for understanding
 - Bottom up or Top Down

- Management
 - How to have external teams do unit testing in a global environment?
 - What documentation is missing from the development phase for effective future testing
 - Test Driven Development advantages for immediate coverage
 - Implicit design decisions by the authors
 - Concurrent vs sequential or post testing

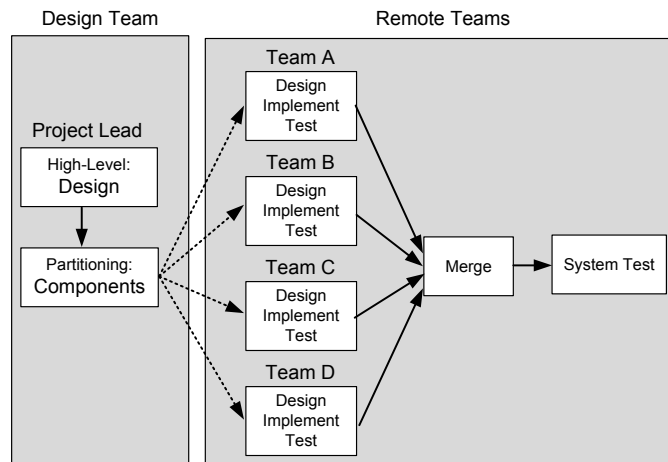
Subordinate Project:



- **Project:**
 - Design team delegates the implementation to the subordinate remote team
 - Remote team implements and tests the supplied design
 - Design team oversees the remote team during this process
- **Implement:**
 - Execution:
 - Centralized Design coming from Group A and implemented in Group B.
 - Then switch it Group B designs, and group A implements
 - Teams do the software design, documentation, implementation, and testing
 - Schedule and manage the deployment of tasks
- **Learn:**
 - Documentation of Architecture
 - UML, Diagrams, Design documents
 - Specifications of the design - Minimum vs too much detail
 - Architecture
 - Big ticket items
 - Decoupled systems, orthogonality, design by contract, refactoring
 - Steering critical design issues vs non-critical issues
 - Performance and resource side effects
 - Support
 - Confusion, support for missing documentation or understanding
 - Asynchronous communication support
 - Understanding external teams background / ability
 - Management
 - Inefficient use of time
 - Over engineered vs not enough
 - Design misunderstandings / Support on designs

- Improving Return of Investment (ROI)
 - Reverse problem
 - Have implementation team to the design documents
 - Design team validate their assumptions and understandings
 - Less communications, better offloading efficiency

Partitioning Project:



- **Project:**
 - Design team breaks a large project into discrete components
 - Assigned to remote teams
 - Responsible for their respective design and implementation.
 - Once all the components are implemented and debugged,
 - Merged into a solution
- **Implement:**
 - Teams do the software design, documentation, implementation, and testing
 - Schedule and manage the deployment of tasks
- **Learn:**
 - Design Partitioning
 - How to break up the design for parallel development
 - Minimize coupling between modules
 - Integration
 - Broken code check-ins to version control
 - Right Hand / Left hand issues
 - Interfaces not coordinated or at fault
 - Discovered at merge time
 - Iteration for success
 - Building streamlined end to end
 - Always working code base and unit tests
 - Management
 - Effective partitioning of the project is difficult
 - Need agreed on vision and design
 - Need leadership to corral outliers
 - Not my problem

Business plan and deployment – Individual Paper Project

----- **SAMPLE:** -----

Imagine that you are a project manager in charge of a new global software project (located in Chicago). Make up a project (your choice) that takes approximately 6 months to complete. The company you are working with is located in China with a 12 hour time zone difference. Assume that the Chinese software developers have equivalent talent as your local staff and they can speak and write fluently in English. You, as the project manager, have control over all 10 software developers, 5 local and 5 in China.

Describe how you would setup and execute a 6 month global software project.

- Please give a speculative budget, travel, infrastructure costs, startup, servers, required software
 - Chinese programmer monthly rates (4K US/month)
 - US programmer monthly rates (8K US/month)

Discussing of development issues:

- Describe the vetting process
 - Budget
 - How do you evaluate the local talent
 - How do you evaluate the local management
- Please describe how you would have the teams setup
 - Communication tools
 - How is it organized?
 - Reporting structure
 - Sub-teams, collective, flat, matrix
 - Version control issues
- Quality control
 - How would you support and control designs?
 - How would you control the quality of work?
- How frequently would your teams communicate between the locations?
 - Individually, group
 - Email, wiki, forums, Skype, etc...
- Training
 - Mentoring on projects
 - Ramping members up on your domain / existing software values
 - Turnovers / Retraining during the course of the project
- Scheduling
 - Dealing with problems and expectations
 - Type of process software process used
- Justification
 - Full budget of this project vs local implementation
 - Contracting services

Changelist Comments

- Everyone is **expected** to submit detail Perforce changelists comments
 - For every assignment 5-100 individual changelists are expected with detailed comments.
 - Please describe in a high level with a few bullet points the task being submitted.

Piazza Discussion forum

- Statistics show: students who participate more and help other students **do better!**
 - The correlation is ridiculous!
 - Poor understanding / poor participation.
 - Great understanding / Great participation
 - As you master the material, help others learn!
 - You're in the master's program so master it!
- Everyone is **expected** and encouraged to participate on the Piazza discussion forum. All class-related discussion here this term.
- The quicker you begin asking questions on Piazza (rather than via emails), the quicker you'll benefit from the collective knowledge of your classmates and instructors. I encourage you to ask questions when you're struggling to understand a concept.
- All correspondence that is not personal in nature should be vectored through Piazza
- Sensitive material, use Piazza private note, not email.
- Keep the forum professional and positive, help each other out.
 - Karma really pays off here.
 - Help each other whenever you can.
 - There will be a section where you'll need help (trust me).

NOTE: Do **NOT** post until you have watched the entire lecture **FIRST** (in class or online)
This will prevent frustration on all sides (members asking or answering questions)

Perforce Submissions

- Every is expected to submit **at least 10 submissions** a week to perforce.
- The biggest reason students get into trouble with software design:
 - Not working on the material frequently enough
 - Taking too large of a bite of the design
- Both are fixed with this **Perforce RULE**
- Even my simplest programs take 10-20 submissions.
 - For these project assignments my average is 40-400 submissions, so 10 will be no problem.

Collaborating together on programming assignments

- You are encourage to work together
 - Use the Piazza forums heavy
 - Even share your material with others in the common directory

- Everyone is 100% responsible for the work they do.
 - If you get help with a section of code,
 - Please refactor the code the **snot out of it**
 - Comment and understand that material
 - Transform the code to **make it yours**
 - Be able to answer **any** question regarding the code you commit
- System for Detecting Software Plagiarism
 - We will be using MOSS - Measure of Software Similarity (Stanford University)
 - Indicates possible code infringements (plagiarism)
 - MOSS - will detect the similarity independent of naming convention, indentation style or formatting, it compares abstract syntax tree of your code.
- If you gain significant support / help from another student
 - Fully disclose the support / help you had in a Readme.txt file submitted with your assignments.
 - Disclosing the help, is **not permission** for copying the code.
 - Only there to clarify and acknowledge help you were given from a fellow student.
- Modifying any Unit Test to alter the outcome results is also an **Academic Integrity Violation**
- If you are stuck and find yourself even tempted to plagiarize
 - Ask for help !!!!
 - Use on Piazza,
 - Visit during offices hours, make an appointment
 - **Don't ever compromise your integrity!**

Tentative Schedule

Date	Lecture	Activity
Week 1 10-Sep	* Class Overview * High-Level Overview * Perforce Demo	Environment
Week 2 17-Sep	* Merging * Branching / Versioning * Maintainability	Perforce Branching
Week 3 24-Sep	* Documentation * Expectations / Deliverables * Agreements	Project 1: Testing Part A
Week 4 1-Oct	* Social * Cultural * Conflict Resolution / Expectations	Project 1: Testing Part B
Week 5 8-Oct	* Design * UML / Diagrams / Design Docs * Architecture	Project 2: Subordinate Role Part A
Week 6 15-Oct	* Communication * Meetings * Multimedia tools	Project 2: Subordinate Role Part B
Week 7 22-Oct	* Scheduling * Agile * Conformance to standards	Project 3: Partitioning Part A
Week 8 29-Oct	* Infrastructure * Version Control / IP / LAG * Equipment	Project 3: Partitioning Part B
Week 9 5-Nov	* Business * Cost / Budget * Offshore, near, on	Project 4: Continuous
Week 10 12-Nov	* Discussion Continuous Project * Justification of remote	Project 5: Business Plan
Week 11 (Finals) 19-Nov	* Final Exam * Business Project due	

September 16, 2014 Last day to add classes to AQ2014 schedule
 September 23, 2014 Last day to drop classes with no penalty, Last day to select pass/fail option
 September 24, 2014 Grades of "W" assigned for AQ2014 classes dropped on or after this day
 September 30, 2014 Last day to select auditor status
 October 28, 2014 Last day to withdraw from AQ2014 classes

Course Policies

Changes to Syllabus

This syllabus is subject to change as necessary during the quarter. If a change occurs, it will be thoroughly addressed during class, posted under Announcements in D2L and sent via email.

Online Course Evaluations

Evaluations are a way for students to provide valuable feedback regarding their instructor and the course. Detailed feedback will enable the instructor to continuously tailor teaching methods and course content to meet the learning goals of the course and the academic needs of the students. They are a requirement of the course and are key to continue to provide you with the highest quality of teaching. The evaluations are anonymous; the instructor and administration do not track who entered what responses. A program is used to check if the student completed the evaluations, but the evaluation is completely separate from the student's identity. Since 100% participation is our goal, students are sent periodic reminders over three weeks. Students do not receive reminders once they complete the evaluation. Students complete the evaluation online in [CampusConnect](#).

Academic Integrity and Plagiarism

This course will be subject to the university's academic integrity policy. More information can be found at <http://academicintegrity.depaul.edu/>. If you have any questions be sure to consult with your professor.

Academic Policies

All students are required to manage their class schedules each term in accordance with the deadlines for enrolling and withdrawing as indicated in the [University Academic Calendar](#). Information on enrollment, withdrawal, grading and incompletes can be found at: cdm.depaul.edu/enrollment.

Students with Disabilities

Students who feel they may need an accommodation based on the impact of a disability should contact the instructor privately to discuss their specific needs. All discussions will remain confidential. To ensure that you receive the most appropriate accommodation based on your needs, contact the instructor as early as possible in the quarter (preferably within the first week of class), and make sure that you have contacted the Center for Students with Disabilities (CSD) at: csd@depaul.edu.

Lewis Center 1420, 25 East Jackson Blvd.

Phone number: (312)362-8002

Fax: (312)362-6544

TTY: (773)325.7296

Retroactive withdrawal

This policy exists to assist students for whom extenuating circumstances prevented them from meeting the withdrawal deadline. During their college career students may be allowed one medical/personal administrative withdrawal and one college office administrative withdrawal, each for one or more courses in a single term. Repeated requests will not be considered. Submitting an appeal for retroactive withdrawal does not guarantee approval. Information on enrollment, withdrawal, grading and incompletes can be found at: <http://www.cdm.depaul.edu/Enrollment-Policies.aspx>