10th Workshop
"Software Engineering Education and Reverse Engineering"
Ivanjica, Serbia

EMULATING REAL-LIFE IN STUDENT PROJECTS IN SOFTWARE ENGINEERING

Vedran Ljubović
Faculty of Electrical Engineering Sarajevo
vljubovic@etf.unsa.ba
Ivanjica, 7. 9. 2010

Introduction – New SE course in Sarajevo!

- 6th semester of Bachelor study
- Capstone course
- Largely based on Somerville
- Lectures + Workshop + Semester-long team projects

Student previous knowledge

- Programming Techniques, 2nd sem. (OOP, C++)
- Software Development, 3rd sem. (OOP cont., frameworks, IDEs)
- Object Oriented Analysis and Design, 4th sem. (UML)
- Basics of Database, 4th sem. (SQL)
- Information Systems, 5th sem. (project mgmt, requirements)
- Software Reliability and Quality Control, 6th sem. - electory (QA, testing)

Rough outline of the course

- Week 1: Introduction
- Week 2-4: Requirements (workshops on technologies to be used)
- Week 5-6: Analyses and design
- Week 7-9: Software development (break in week 8)
- Week 10-11: “Change management”
- Week 12: Testing
- Week 13-14: Testing (cont.), integration, documentation review

Grading

- 10 points - attendance
- 10 points - homework
- 20 points - team project
- 20 points - written exam (theory)

- 40 points - oral exam (project & theory)

Total: 100 points

Workshops

- Introduction to technologies used
- Issues needed for projects that are not covered in Somerville (in sufficient detail)
- Project administration issues
- Literature: Pressman, McConnel

Projects

Team projects - goals

- Learn to work as a team with diverse personalities
  - Project management & planning skills
- Learn real-life implications of SE issues
- Use CASE tools:
  - Redmine http://www.redmine.org
  - use of IDE wasn't mandatory, but it made life much easier; all teams used either Eclipse or Netbeans

Team projects - goals (cont.)

- A minimal guidance approach
- The goal of project was to fail!
  - ...and learn from mistakes
  - use waterfall method

Week 1: Project kickoff

- Students are divided into 4 member teams - alphabetically
- Projects are ill defined
  - single vague sentence, such as:
    "Build a system for elevator control in a corporate building."
  - no further instruction
- Only requirement: use Java!

Weeks 2-4: Requirements gathering

- Students are on their own, did a fairly good job
- Interview
  - Interviewee is **not** the manager
  - Interviewee: a) has insufficient/wrong information, b) can't make decisions, c) doesn't care for project success
- Customer doesn't sign the SRS

Weeks 5-6: OOA&D

- No specific instructions - “do whatever you think you'll need”
- Result was: poor design
- Change event: new requirements

Weeks 7-9: Software development

- Supposed deadline is May 1st
  - week 7 started at April 5th
- Interrupted by exams (week 8)
- Prototype required
  - but students weren't told that it's a prototype - 2nd mistake

Vedran Ljubović,
Emulating Real-life in Student Projects in Software Engineering
Ivanjica, 6.9.2010

Weeks 10-11: The Great Reshuffle

- No student on the same project, no two students on the same team
- Evaluate each others documentation and code
- Learn to adapt to code developed by other people

The Great Reshuffle (cont.)

- Break up existing division of work patterns
  - coders were teamed with coders, designers with designers...
- Followed by a major change event
  - cascade from SRS, through design docs to code
- Evaluation results
  - some teams did all work off-repository, panicked, pulled all-nighters; “old teams” worked with “new teams” to get favorable evaluation

Weeks 12-13: Testing

- Teams were required to build unit tests
  - proved difficult due to poor architecture (all code crammed into GUI events); teams had to do extensive refactoring to make unit testing possible
- “Test teams” were formed
  - no student was testing a project he ever worked on before
- Bugs were reported using Redmine
  - “bugs” include documentation and packaging issues

Weeks 12-13: Testing (cont.)

- Testing in action
  - students gained points for reporting a bug, lost points for not fixing a bug
  - but not if the bug was rejected by development team
  - each fix must be linked to a SVN commit
  - each bug rejection must be heavily documented
  - arguments and fights (!?) ensued

Week 13: Documentation

- Overlap with testing stage
  - first week of testing included just SRS and design documents, second included (just written) documentation

- Had to give students an elaborate list of documents to prepare
  - user documentation: user manual, help, context-sensitive help, release notes, install instructions
  - technical documentation: build requirements and instructions, guide to the code tree, naming policies etc.

Weeks 13-14: Project review

- Did a thorough review of projects
- Key issues found
  - some design documents weren't updated after change events
  - SRS contains features NOT requested by user, which weren't implemented (over-ambitious SRS)
  - some projects unfinished
- Students had to fix those issues in remaining time

Week: 14: Integration

- All projects have to interoperate according to given specification
  - students to negotiate technical details
- Only a week of time to do it! (3rd mistake)
- Failed completely

Final grading

- Each team from first half was awarded up to 10 points, likewise for second half
- Each students points were sum of points for his/her first team and second team
- A few exceptions were made

Final results and project testing

- 10 teams of 12 delivered a fully working product
  - of those, 4 teams neglected some explicit user requirement(s)
  - one team had to work from scratch and didn't make it in time
- Of 48 students none failed the project part (but 9 passed just barely)
  - one student never showed up or did anything! he failed of course

Student feedback

- "Project was too hard"
  - "Course has too few ECTS" (currently 6,0)
  - "Project carried too few points"
- "Project was frustrating"
  - "This is not how things are in real-life!"
  - "We are not smart enough to learn from mistakes"
- "Mr. Cowboy Coder didn't let me do my work"
- Disconnect between lectures and project activities

Conclusions

- What is "real-life"?
- Issues with project size / team size / student load
- When and how to apply guidance?
- "Change events" are useful