Agile Processes in Software Engineering - An Educational Perspective

Dr. Jean-Guy Schneider
Swinburne University of Technology
jschneider@swin.edu.au
Swinburne University of Technology

- Technical University, founded in 1990 (merger of two technical colleges)
- 6 Campuses in metropolitan Melbourne, 2 Campuses overseas (Sarawak, Tummasiri)
- ~9’500 Higher Education students
- IBL for most Higher Education Courses
School of Information Technology

- ~70 staff members (5 professors, 42 lecturers, 25 admin/technical)
- 4 research centres:
  - Swinburne Computer-Human Interaction Laboratory (SCHIL)
  - Centre for Internet Computing and E-Commerce (CICEC)
  - Centre for Molecular Simulation (CMS)
  - Centre for Software Engineering (CSE)
- 6 undergraduate degrees (CS, SE, IS), various postgraduate degrees (Masters, Ph.D., Graduate Diplomas etc.)
- ~1’800 students (full- and part-time, 50% undergraduate), 25%-30% international
Overview

- A Personal Note
- Goals of Software Engineering Education
- Principles of Agile Development Processes
- Teaching Agile Processes in Tertiary Education:
  - Pair Programming
  - Working Environment
  - Motivation, Participants Goals
  - Limitations of Agile Processes
- Teaching in Industrial Environment
- Concluding Remarks, Open Questions
What do we want/have to teach Software Engineering students at a tertiary institution?
Software Engineering Body of Knowledge (Swebok)

- Software Requirements
- Software Design
- Software Construction
- Software Testing
- Software Maintenance*
- Software Configuration Management*
- Software Engineering Management
- Software Engineering Process
- Software Engineering Tools and Methods*
- Software Quality

http://www.swebok.org
### IS Perspective on Tertiary Education

<table>
<thead>
<tr>
<th>Desired Characteristics of Graduates</th>
<th>Rank Business</th>
<th>Rank University</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication skills</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>Capacity to learn new skills and procedures</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>Capacity for cooperation and teamwork</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>Capacity to make decisions and solve problems</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Ability to apply knowledge to workplace</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Capacity to work with minimum supervision</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Theoretical knowledge in a professional field</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>Capacity to use computer technology</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>Understanding of business ethics</td>
<td>9</td>
<td>12</td>
</tr>
<tr>
<td>General business knowledge</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>Specific work skills</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td>A broad background of general knowledge</td>
<td>12</td>
<td>10</td>
</tr>
</tbody>
</table>
2001 Survey of School of IT

Survey on current status of software engineering practice and research.

- How well does current practice conform to an ‘engineering’ discipline?
- What are the main current issues facing real software developers?
- How should they drive the research agenda?
- What are the main issues for educators?

Answers from 28 academics working in software engineering in Australia.
In your opinion, what are the three most pressing issues facing mainstream software developers today?
Question 3: Response from Academics

[Bar chart showing frequency of categories such as Software quality, On time / in budget, Requirements, Time to market pressure, Maintenance, Lack of standards, Components and reuse, Estimation, Employee issues, Development process, Speed of technology evolution, Methods that scale up.]
Question 7

In your opinion, what are the most important facets of software engineering that you believe a university degree in software engineering should focus upon?
Question 7: Response from Academics
Educational Goals

- Software development as engineering activity,
- Competence in a variety of technologies and methods suitable for large-scale systems,
- Importance of quality assurance
  - Validation & Verification, traceability, etc.
- Skills for development in larger teams,
- Working with little supervision,
- Communication skills (incl. technical writing),
- Acquiring knowledge and skills in projects with real clients.
A Student Perspective

I think our software engineering subjects are missing the hands on component. At the moment they seem to consist of lectures explaining various parts of documentation (design documentation, requirement documentation, etc.). Software engineering subjects tend to forget that the key motive to software engineering is to develop better software. If the students can put what they learn into practice, I am sure they will not only find the subjects more interesting, but also achieve better results.

Swinburne IT student
Opportunity for Change

- There is a need/opportunity to improve current SE education practices.

- Can we make software engineering education more practice oriented (and less “document centred”)?

- Can we adopt some of the current industrial software development practices?
  - If so, what are the benefits/consequences?

- Can we focus more on the human aspect of software development - who are the “users” of development processes?
Should we introduce the practices of agile development processes into tertiary education?

If so, how can we introduce the practices of agile development processes into tertiary education?
eXtreme Programming Practices

- Coding standards
- Pair Programming
- Testing
- Small development steps
- Continuous integration
- Small, frequent releases
- Simple design
- Refactoring
- On-site customer
- “Planning Game”
- 40-hour week
- Metaphor
- Collective Ownership
- Continuous learning
- Stand-up meetings
The good news first...

- Importance of testing
- Focus on quality, coding standards
- Cost vs. time vs. quality vs. scope
- Configuration management
- Software developers are valuable resources

Better understanding and acceptance due to the ‘eXtreme’ metaphor!
Pair Programming

- Probably the most investigated practice:
  - Several experience reports published, encouraging results in programming courses.

- But:
  - Influence of unbalanced knowledge in pairs,
  - Steep learning curve before PP is effective,
  - Requires understanding what programming is all about!

- Main motivation of many students: pass the subject!
  - How to motivate them to tutor their partner?
  - How to ensure real teamwork?

- Assessing work done in pairs?
- Gender and personality issues in pair programming.
Process vs. Discipline

Agile = less process, more discipline
Students like less process, but they like even less discipline.

But: any agile development process needs to be applied in a very disciplined way, otherwise disaster is very likely.

A less flexible development process with more guidance better suited for teaching purposes!?
Collective Ownership

- Students not familiar with ego-less culture of collective ownership:
  - Often a matter of age and maturity,
  - Cultural backgrounds play an important role.
- Changes have to be made in a disciplined way
  - Students do not fully understand consequences.
- Importance of risk management:
  - How to restore “undisciplined” changes?
  - Highlights importance of conf. management!
Working Environment

- Lab rooms enabling (or even encouraging) pair programming:
  - General rule: as many computers into a room as possible.
  - Dedicated rooms for projects difficult (if not impossible) to get.
  - Students work at home (alone) instead of in small groups!

- Scattered timetables makes it difficult to organize joint activities outside scheduled classes:
  - Disables effective face-to-face communication,
  - Need for stricter rules to document activities.

- Having an on-site customer for cap-stone projects is not realistic!
  - Other means for getting timely feedback required.
Agile Cost Model

Cost of change

Source: Kent Beck, eXtreme Programming Explained, 1999
Refactoring

- Refactoring is one of the key practices in agile development:
  - If it is not done regularly, a project will soon be back on an exponential cost curve!

- Requires a certain level of experience why, when, and how to refactor.
  - “easy” to teach how (and probably why),
  - when is a different matter!

- Tool support often essential, automated test suites can help considerably.
Project Deliverables

We should have chosen an XP process, then it would not have been necessary to write a requirements specification.

Swinburne IT Student

The most efficient and effective method of conveying information to and within a development team is face-to-face communication.

Agile Alliance Manifesto
Project Deliverables (cont.)

- Running software is considered to be the most important deliverable of a project
  - What about other deliverables?
  - What about maintenance? Large-scale systems?
  - Non-documented artefacts are very volatile!
  - Difficult to assess students based on software alone.

- Working software is the primary measure of progress
  - In a learning environment, does this still hold?
  - Lessons learned are of equal/greater importance!
Team Member Selection

- Agile development requires experienced developers:
  - *Get the best developers you can!*
  - *Agile development is suitable for a group of people who know what they are doing.*
  - *XP is for professionals, not students.*

- How to select suitable students?
  - “Bad” students cannot be replaced!

- Mentoring (importance of a team coach):
  - Problem to get suitable/experienced mentors in tertiary education.
Communication Skills

Recent survey of software projects revealed that

- software developers often refrain from using new development technologies:
  - *The people on the projects were not interested in learning our system.*

- ad-hoc techniques can be applied successfully:
  - *They were successfully able to ignore us, and were still delivering software, anyway.*

♫ Good communication is the most significant single factor for success.

Communication Skills (cont.)

- Students may elect software engineering studies because they do not want to work in teams.
  - Good communication skills important in any SE project, but essential for agile development.

- Subjects on team organization not part of Software Engineering curriculum.
  - Individual assessment hinders development of communication skills!
Limitations of Agile Processes

- Developing large, complex software
- Safety-critical software, real-time systems
- Clients vs. users, adequate user-interfaces
- Developing reusable artefacts
- Distributed development
- Development involving large teams
- Subcontracting

Issues tertiary education should address!
Teaching XP in Industrial Environments

- Pair programming
- Collective ownership
- Working environment, on-site customer
- Refactoring
- Project deliverables
- Team member selection, expertise
- Process vs. discipline
- Communication
Conclusions

- Little experience in teaching agile development processes at tertiary institutions
  - More experiments needed to gain further insight.

- Some practices of agile development contradict educational goals, others are well-suited for teaching.
  - SE curricula need to be focussed accordingly.

- Applying agile development practices requires experience:
  - How do we “teach” experience?
Open Questions

- Which technical skills are essential, which ones desirable?
- Suitable environments to enable collective learning experience?
- How to improve communication skills?
- Agile development and online learning?
- Agile development in a distributed environment?
- ...

CHOOSE SIG Beer, June 18, 2002
Your questions?