A Cross-Program Investigation of Students' Perceptions of Agile Methods

Grigori Melnik
University of Calgary
2500 University Drive NW
Calgary, Alberta, T2N 1N4, Canada
+1-403-210-9710
melnik@cpsc.ucalgary.ca

Frank Maurer
University of Calgary
2500 University Drive NW
Calgary, Alberta, T2N 1N4, Canada
+1-403-220-3531
maurer@cpsc.ucalgary.ca

ABSTRACT
Research was conducted on using agile methods in software engineering education. This paper explores the perceptions of students from five different academic levels of agile practices. Information has been gathered through the collection of quantitative and qualitative data over three academic years, and analysis reveals student experiences, mainly positive but also some negative. Student opinions indicate the preference to continue to use agile practices at the workplace if allowed. A way these findings may potentially be extrapolated to the industrial settings is discussed. Finally, this report should encourage other academics considering adoption of agile methods in their computer science or software engineering curricula.

Categories and Subject Descriptors
D.2.9 [**Software Engineering**]: Management – *Life Cycle, Productivity, Programming Teams, Software Process Models.*

General Terms
Management, Performance, Design

Keywords
Agile methods, eXtreme Programming, empirical study, perception analysis

1. INTRODUCTION
A fleet of emerging agile methods of software development (with extreme programming being the flagship), once considered a novelty, is now moving towards the mainstream of the software industry. In a nutshell, agile methods are human-centric bodies of practices and guidelines for building usable software in unpredictable, highly-volatile environments. Essentially, all agile methods encourage continual realignment of development goals with the needs and expectations of the customer. They concentrate on significantly improving communications and interactions (among team members and with the customer), promote continuous feedback, focus on "clean code that works", transparency, and merciless testing to achieve higher quality.

As the ideas of agile methods increase in popularity, a controversy around them continues to grow. One of the problems is that most of the evidence of agile methods effectiveness available is anecdotal. Real-world examples argue for (see, for instance, [1], [5]) and against ([12]) agile methods. Several leading software engineering experts suggest that finding "home grounds" and, perhaps, synthesizing, "balancing" the two (agile with tayloristic) may provide developers with a comprehensive spectrum of methods ([2], [3], [7]).

On the way of "crossing the chasm", agile methods are slowly entering academia and becoming part of the computer science and software engineering curricula. Importantly, the newest IEEE/ACM Computer Science – Software Engineering Curriculum lists agile concepts and several practices (e.g., refactoring, test-driven development) as essential topics [8]. For a comprehensive literature review of cases/studies supporting or challenging agile practices in software engineering curriculum, the reader is referred to Section 1 of [9].

We have been introducing agile methods in software engineering courses at the University of Calgary and Southern Alberta Institute of Technology since 2001. Perceptions of broad student body on agile methods in general and individual development practices were studied. Earlier studies ([9],[10]) concentrated mainly on the qualitative analysis. A discussion on why agile methods should be taught together with our lessons learnt and the recommendations to other academics thinking of introducing agile methods in their courses are contained in [9].

This paper not only updates the results of the three-year long study but also contains the detailed analysis of the aggregated quantitative data (including both perceptions and the related associations).

2. STUDY OVERVIEW
The intent of our study is threefold: 1) to explore the perceptions of agile practices from the students' perspective; 2) to examine associations between students' perceptions of agile methods in general and their perceptions of individual practices (pair programming, project planning using the planning game, and test-driven development, a.k.a. test-first design); 3) to investigate how students' perceptions vary (if at all) depending on the academic

ICSE'05, May 15–21, 2005, St. Louis, Missouri, USA.

program, age and years of prior industry experience. The study focuses on agile engineering practices that are coming from eXtreme Programming (XP). Note that we are using the expression "agile practices" to make it clear that we did not use the full set of XP practices in our study (for the list of applied practices per course see column "Practices" of Table 1).

The study instrument, a questionnaire, consisted of 20 items which included both quantitative questions (on a 5-point Likert summated scale, 1 "strongly disagree" to 5"strongly agree") and qualitative open-ended and Questions assessed students' perceptions of the agile practices and also gathered their suggestions on how the courses can be improved.

When looking at the students' experiences, we asked a number of questions:

- Did the students enjoy working on the projects using agile practices?
- What worked for them?
- What problems did they encounter?
- Whether students believe that using XP improves the capabilities of small development teams (code quality, productivity)?
- Whether they would use agile practices in the future (if allowed) or not?
- How did XP improve their learning?

Figure 2 contains the complete listing of quantitative questions and the responses. The questionnaire was executed on the Web. Students were given one week to anonymously respond.

In addition, interviews and discussions were conducted during the course of the semester to get some informal feedback on other aspects of agile methods that students were exposed to during the courses (continuous integration, collective code ownership, refactoring, coding standards).

Table 1. Programs and Courses.

Abbreviation	Academic program	Institution	Course	URL(s)	Instructor (Semester)	Practices	Team size
DIPL	College-level Diploma	SAIT	Data Abstraction & Algorithms (CS2)	http://webctce.sait.ca/public/cmpp307_melnik/	Author 1 (F01, W02) Non-author (F03)	*CS, CUST, PP, R, TEST, TDD, SR, YAGNI.*	2
POSTDIPL	College-level Post-Diploma Applied Bachelor's Degree	SAIT	Software Testing & Maintenance	http://webct.mysait.ca/script/M_APSE502/scripts/serve_home	Author 1 (F02, F03)	*CCO , CI, CS, CUST, PG, PP, R, TEST, TDD, SR, YAGNI*	2 – 4
			Internet Software Techniques	http://mase.cpsc.ucalgary.ca/EB/Wiki.jsp?page=Root.APSE504w04	Author 1 (W02, W03, W04)	*CCO , CI, CS, CUST, M, PP, R, TEST, TDD, SR, YAGNI*	3 – 5
			Software Engineering Project	http://mase.cpsc.ucalgary.ca/apse503	Author 1 (W03,W04)	*CCO , CI, CS, CUST, M, PG, PP, R, TEST, TDD, SR, YAGN.*	6 – 9
JUNIOR	University-level Junior Undergraduate	U of Calgary	Foundations of Software Engineering	http://sern.ucalgary.ca/courses/cpsc/333/w03/	Non-author (W03)	*CI, CS, CUST, M, PP, R, TEST, TDD, SR, YAGN.*	6 – 8
			Principles of Software Engineering	http://sern.ucalgary.ca/courses/seng/311/w04/	Non-author (W03,W04)	*CI, CS, CUST, M, PP, R, TEST, TDD, SR, YAGNI*	6 – 8
SENIOR	University-level Senior Undergraduate	U of Calgary	Web-Based Systems	http://mase.cpsc.ucalgary.ca/EB/Wiki.jsp?page=Root.SENG513w04	Author 2 (F02) Author 1 (W03, F03, W04)	*CCO , CI, CS, CUST, M, PG, PP, R, TEST, TDD, SR, YAGN.*	4 – 7
GRAD	University-level Graduate	U of Calgary	Agile Software Engineering	http://sern.ucalgary.ca/courses/CPSC/601.93/F2003	Author 2 (W02, F02, F03)	*CCO , CI, CS, CUST, M, PG, PP, R, TEST, TDD, SR, YAGNI* + Scrum	6, 11

Note 1: Abbr. F01 = Fall 2001, W02 = Winter 2002 etc.

Note 2: Abbr. *CCO* = Collective Code Ownership, *CI* = Continuous Integration, *CS* = Coding Standards, *CUST* = On-site/On-call/Online Customer, *M* = Metaphor/Architecture, *PG* = Planning Game, *PP* = Pair Programming, *R*=Refactoring, *TEST* = Unit Testing, *TDD*=Test-Driven Development, *SR* = Short Releases, *YAGNI* = "You Ain't Gonna Need it" (Simple Design).

3. COURSES AND STUDENT POPULATIONS

The sample employed in this study consisted of 240 volunteers out of 693 students who were asked to participate. These were students from five different levels of computer science programs at the Southern Alberta Institute of Technology (SAIT) and the University of Calgary. Ages of the students ranged from 19 to 46 years (mode = 22, median = 27). All individuals were knowledgeable about programming. The distribution of respondents by programs and years of industry experience is given in Table 3. Data was collected partially during and partially at the end of academic semesters in which agile practices were introduced. Data collection was anonymous and the instructors (including authors of this paper and other instructors) were not able to determine who participated in the study and who wrote specific comments.

Table 1 outlines course and program characteristics. Detailed descriptions of the programs, courses, environments, course projects and tools can be found in Sections 3.1–3.5 of [10]. In all courses with the exception of DIPL and JUNIOR, students were introduced to the agile practices at the very beginning of the course. The values of communication, simplicity and feedback were strongly emphasized. As can be seen from Table 4, vast majority of students in DIPL, POSTDIPL, and JUNIOR programs were unfamiliar with agile methods. SENIOR students acknowledged "somewhat familiarity", unlike SENIOR and GRAD groups were majority of students were familiar with agile methods. This level of GRAD group interest in agile methods can be by the fact that the course is not required for completion of M.Sc. degree and, therefore, students taking this course were interested in agile methods (with two individuals already having prior industrial experience with XP).

We would also like to point out that students do not work full time on a course. We estimate that on average a student spends about 5-7h/week on the course assignment.[1] Hence, the effort going into a single iteration is approximately 20 hours per student (which is much lower than in XP or any other agile method).

4. FINDINGS

The average response rate is 55% among SAIT students, 24% among University of Calgary undergraduate students (note a significantly larger population), and 83% among University of Calgary graduate students (Table 2).

Figure 1 (answers shown by the academic program) illustrates that the overwhelming majority of all respondents (78%) either believe or strongly believe that using XP improves the productivity of small teams. Seventy-six percent (76%) suggested that XP improves the quality of code and 65% of all respondents would recommend XP to the company they work for or may be working in the future. Note that in question formulation, we did not explicitly specify the baseline for comparison and left these questions open for interpretation by the respondents (assuming that they would be comparing quality and productivity aspects with their prior experiences with any non-agile process).

[1] This estimate is based on time sheets over 10 weeks from one of the GRAD groups.

Table 2. Summary of Respondents by Academic Programs

Academic program	Semester(s)	# of invitations sent out	# of respondents	Response rate
DIPL	Fall 2001, Winter 2002	41	25	61%
	Fall 2003	40	9	23%
POSTDIPL	Winter 2002	22	12	55%
	Fall 2002	18	10	56%
	Fall 2003	23	22	96%
	Winter 2004	17	11	65%
JUNIOR	Winter 2003	175	25	14%[2]
	Winter 2004	142	18	13%[2]
SENIOR	Fall 2002	55	19	35%
	Winter 2003	62	19	31%
	Fall 2003	33	20	61%
	Winter 2004	24	16	67%
GRAD	Winter 2002	12	9	75%
	Fall 2002	11	8	73%
	Fall 2003	18	17	94%
Total, All Programs		**693**	**240**	**35%**

We relied on the fact that respondents had a "natural" understanding of such terms as "quality" and "productivity" (many decisions in the industry are also made on such fuzzy notions). In addition, allowing students to individually interpret these terms provided an opportunity for them to take a self-reflective view of their experiences.

Participant responses to all ranking questions are summarized in Figure 2. The results are overwhelmingly positive. This holds for XP in general and for individual practices. It also holds across all levels of students (with graduate students being a little more skeptical; Figure 2 and Table 5). Analysis of the results of Spearman's correlation test revealed that respondents who found the experience of working in agile teams so positive to recommend XP to their companies (current if employed by the software product/service industry or future if not) were also more likely to believe that:

- Using XP improved the quality of code (ρ=0.67, $p<0.0001$);
- Using XP improved the productivity of small teams (ρ=0.67, $p<0.0001$).

Some[3] free-response comments strongly support the above findings:

[2] It is quite possible that this low response rate among junior undergraduate students is attributed to the fact that the survey invitations were sent after the term end.

[3] Detailed qualitative analysis with categorization of students' comments can be found in Section 5 of [10].

Table 3. Distribution of Respondents by Academic Programs and Years of Industry Experience[4]

	none	< 1 year	1-3 years	3-5 years	5-10 years	> 10 years
DIPL	34%	33%	33%	0%	0%	0%
POSTDIPL	27%	9%	28%	15%	12%	9%
JUNIOR	37%	16%	26%	11%	5%	5%
SENIOR	26%	26%	34%	11%	0%	3%
GRAD	6%	17%	12%	18%	12%	35%
Across All Programs	26%	19%	27%	12%	6%	10%

Table 4. Apriori Familiarity with Agile Methods[4]

	Very familiar	Somewhat familiar	Unfamiliar
DIPL	11%	0%	89%
POSTDIPL	12%	55%	33%
JUNIOR	0%	26%	74%
SENIOR	3%	74%	23%
GRAD	29%	53%	18%
Across All Programs	10%	51%	39%

- *"Quality is built into the process (not a supporting concept but a core concept)."*
- *"I believe that XP helps get more work done in less time and is very effective for small groups as it allows for the group members not to get stuck for extended periods of time."*
- *"Focus on results. Focus on small, fast deliverables. Focus on communication. Focus on minimalization. Focus on teamwork. I love it."*

Furthermore, analysis revealed no significant correlation between the participants' age/program enrolled/years of industry experience and beliefs that using XP improves the quality of code and the productivity of small teams (Table 5). The only exception to this is a weak negative correlation between the program/academic level and the perceptions about the planning game. More experienced respondents were less confident with the estimation of user stories (ρ= -0.22, p<0.05) and less likely to believe that using the planning game would make the team more adaptive (ρ= -0.21, p<0.01). This is expected as more mature students are well-aware of the inaccuracies of estimation (based on their personal experiences both at school and at work). The planning game seems to be the least popular practice (out of all practices the quantitative data was gathered on). This can be attributed to the lack of experience in project planning and estimation and the fact that in many of the courses students used technology that was new to them – which makes effort estimation inherently difficult. However, a large number of student comments on the planning game were more positive than the quantitative data leads us to believe:

- *"The planning game resolves misunderstandings, gives a good overview of the path and goal the iteration is following. What worked well is having the customer involved, this gives the team and the customer and good idea of what can be accomplished."*
- *"The planning game gives everyone a good understanding of the requirements. The part that the developer assigned to a user story gives the estimates for the user story is good."*

Students also related to the fact that the planning game helped to steer the project in the right direction and the small releases helped "*to distribute the load more evenly*" and "*to keep the team on track*".

All ten simple correlations between attitudes of individual XP practices and the expressed willingness to recommend and use XP in the future workplace were statistically significant (p<0.0001) and ranged from 0.34 for the agreement that test-driven development improves software design to 0.56 for the level of personal enjoyment of pair programming. The data indicates that students who agreed that using XP improved the quality of code were more likely to believe that this (quality improvement) was partially due to pair programming (ρ= 0.41, p<0.0001) and test-driven development (ρ= 0.41, p<0.0001).

A further examination of the results indicates a weak positive correlation between the age of respondents and attitudes towards test-driven development (TDD), which might be explained by the higher level of discipline of more mature students. Generally, this practice is not easy to put into action because students are not used to thinking the test-first way. We believe that the underlying reason for this is that TDD is not about testing but about design. Afterwards, doing design is hard – independently from how you document it (as test code or in UML). TDD simply forces design issues to the foreground while UML diagrams can be sloppy as they are hard to evaluate by markers in an academic setting. This impression was supported by respondents' comments:

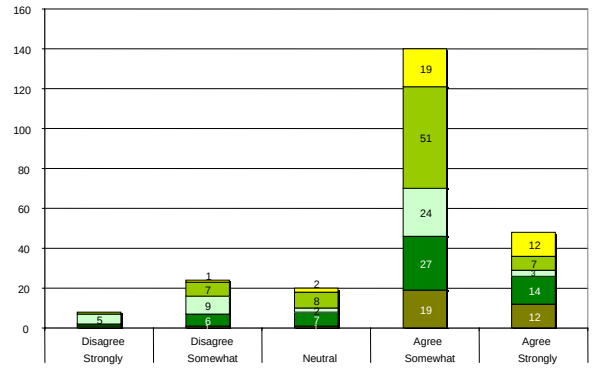
- *"I think the test code is more a part of design then it is just testing."*
- *"I felt that testing first gave a better sense of "here is what must be done", and how to approach it."*

Some of the students believed it was logically confusing and *"almost like working backwards."* They did not know how many tests would be enough to satisfy that the desired functionality would be implemented correctly. Also, some believed testing involved too much work and they did not see the short-term benefits.

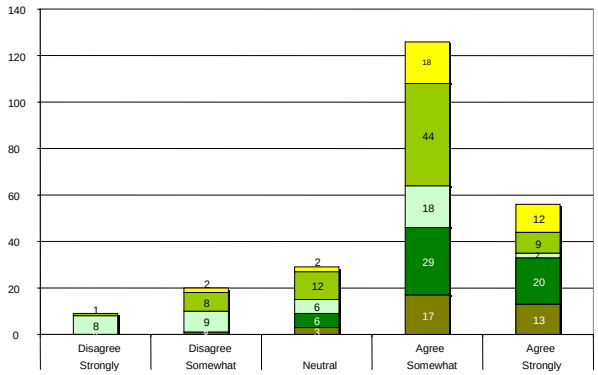
To address some of the problems students were having with test-driven development, we introduced an additional practice (in winter 2004) – user-acceptance testing with FIT[5]. FIT tests are a tabular representation of customer expectations that can be understood by human beings. These tests were used as the primary mode of communicating customer requirements to the

[4] N=113, these statistics reported are only based on the data of fall 2003 and winter 2004 semesters since the earlier data was not collected.

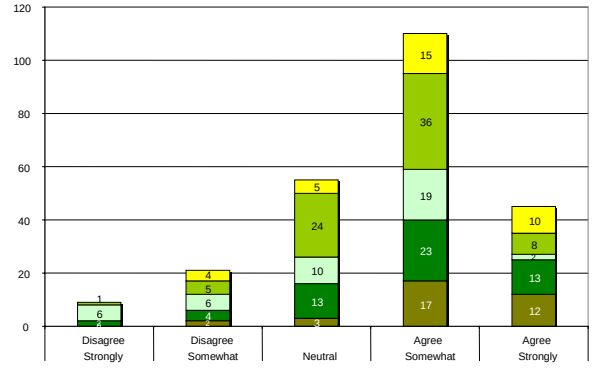
[5] http://fit.c2.com



Q1. I believe that using XP improves the **productivity** of small teams.



Q2. I believe that using XP improves the **quality** of the code.



Q3. I would **recommend** to my company to use XP.

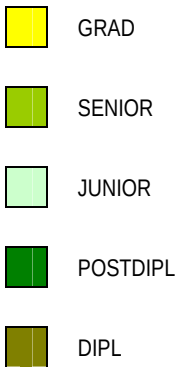


Figure 1. Extreme Programming Perceptions Distribution by Academic Programs (N=240).

students. A well-defined test suite was provided by the customer (instructor) up front. In a separate experiment designed to evaluate the suitability of using FIT for specifying functional requirements for the developers, we have found that these tests can be easily understood, interpreted and implemented by developers [11].

It was also established that among those who would recommend the use of XP or other agile practices to their companies, a large number of respondents personally like pair programming and prefer to work in pairs in the future (if allowed). Subjects felt that doing code inspections by pairing continuously is more efficient than traditional debugging. This finding was statistically significant (ρ= 0.40, p<0.0001) and is contrary to the convention (low popularity of pair programming in industry).

With regard to adapting XP as a development process, we believe it is more difficult to make XP work in the academic environment then in the industrial. This is simply because of scheduling problems (impossible to collocate students every day) and the amount of time a student can spend on the project per week (impossible to get them to work on the project every day). The logistic of the process is trickier and students constantly switch between tasks (i.e. work on different courses). Other challenges that students identified include: scalability and criticality, customer's availability, fixed-price contracts, and developer's abilities – all the issues currently being discussed in the industry.

5. LIMITATIONS OF THE STUDY

As with all empirical studies, threats to the internal and external validity exist. The self-selection of study participants might have skewed the results to the positive and negative extremes. Students with any strong opinion might be more willingly answered the questionnaires compared with students who simply did not care. The overall response rate of 35% might counter that threat. The instructors of the course and their enthusiasm for agile approaches also create a bias. Responses from students in courses taught by the authors are slightly more positive that from classes taught by others. The study focuses on perceptions as the authors believe that these have a substantial impact on actual performance (based on organizational psychology studies that suggest that happy teams are productive teams). We do not have any objective data to validate the correlation between perceptions and actual performance in the process. Our sample is taken from a broad set of students in various educational programs and from different

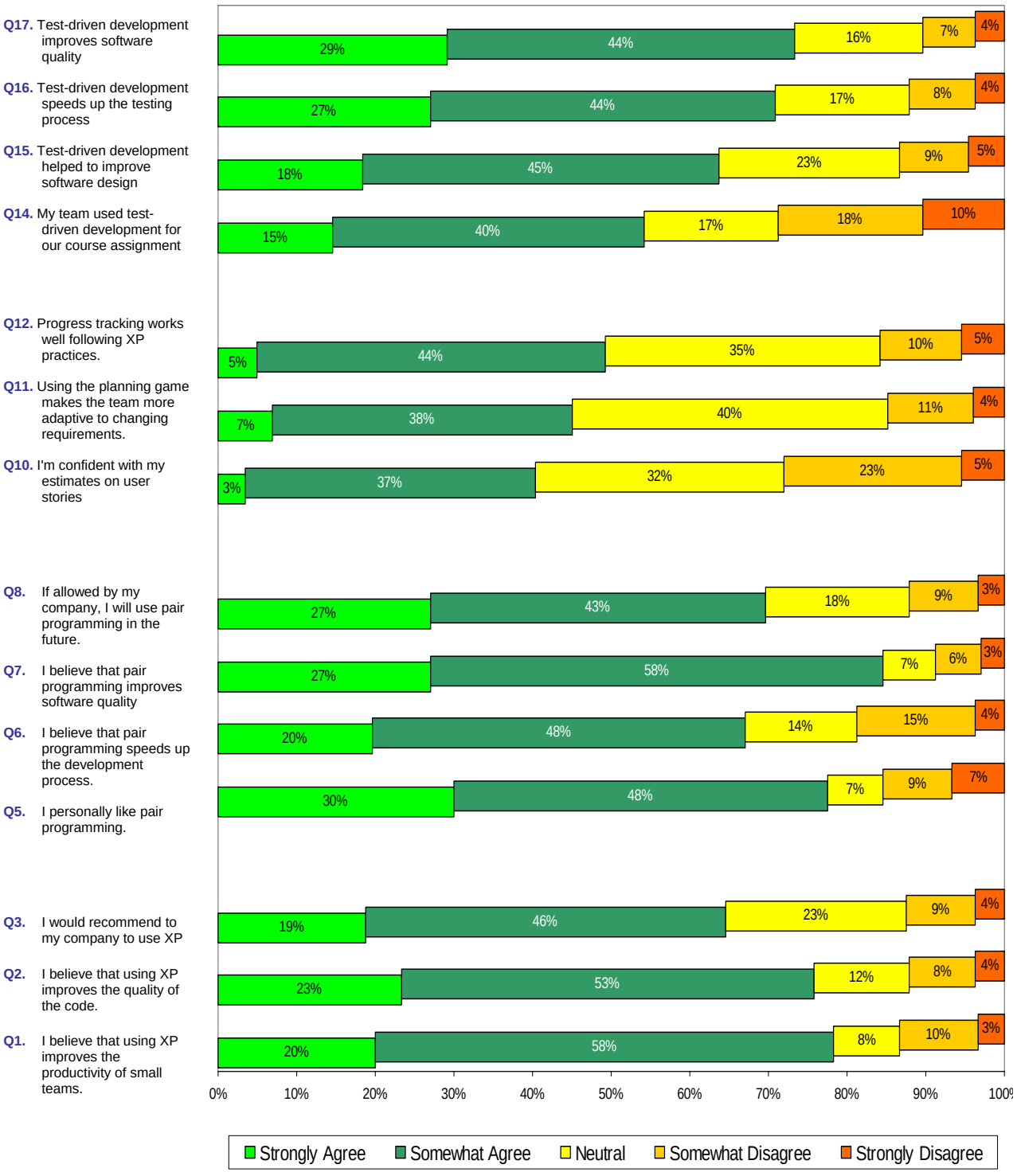


Figure 2. Cumulative Answers of Students from All Programs (N=240).

Table 5. Spearman's Correlations of Items Reflecting Attitudes towards XP with Respondents' Age, Academic Program Enrolled, and Years of IT Experience.

	Age	Academic Level (Program)	Years of Experi-ence	(A)	(B)	(C)
Extreme Programming in General						
Position on whether using XP improves the quality of code **(A)**	0.15	-0.11	0.06	–		
Position on whether using XP improves the productivity of small teams **(B)**	-0.07	-0.17	0.08	**0.63***	–	
Willingness to recommend and use XP in the future **(C)**	0.03	-0.12	0.06	**0.67***	**0.71***	–
Pair Programming						
Level of personal enjoyment of pair programming (PP)	0.03	-0.12	0.08	**0.45***	**0.48***	**0.56***
Agreement that PP speeds up the development process	-0.07	-0.05	-0.03	**0.31***	**0.48***	**0.36***
Agreement that PP improves software quality	0.19	-0.06	0.01	**0.41***	**0.45***	**0.50***
Willingness to use pair programming in the future (if allowed) (N=107)	0.10	-0.08	0.01	**0.40***	**0.48***	**0.49***
Test-Driven Development (TDD)						
Self-reported level of practicing TDD on the course project/assignments (N=106)	**0.26****	-0.13	0.15	**0.38***	**0.36***	**0.48***
Agreement that TDD speeds up the testing process (N=104)	**0.28****	-0.14	0.08	**0.43***	**0.42***	**0.44***
Agreement that TDD helps to improve software design (N=105)	**0.23*****	-0.05	0.16	**0.27****	**0.34***	**0.34***
Agreement that TDD improves software quality (N=104)	**0.25*****	0.00	0.17	**0.41***	**0.39***	**0.47***
Planning Game						
Agreement on whether using the planning game makes the team more adaptive (N=100)	0.11	**-0.21****	0.17	**0.30****	**0.44***	**0.45***
Confidence with estimation of user stories (N=99)	0.18	**-0.22*****	0.15	**0.25****	**0.35***	**0.34***

Note 1: Generally, N=113 (statistics reported are based on the data collected during fall 2003 and winter 2004 semesters only). Values of N <113 are attributed to the exclusion of the cases with the 'N/A' responses.

Note 2: *p<0.0001, **p<0.01, ***p<0.05

years in the educational program. In that sense, we are confident that our sample is a good approximation of student populations. Clearly, we cannot assume that similar results would be seen when sampling from developers working in industry.

Nevertheless, we would like to point out that the results gathered from our graduate student population – who often work full time in industry – are not significantly different from the overall results[6].

[6] There exist other studies looking into usefulness of using academic courses for empirical validation of software development processes in making decisions related to software process improvement (e.g. [4], [6]).

6. SUMMARY

This research has explored and revealed student perceptions of agile methods. Our three year experiences introducing agile methods in the Computer Science courses indicate that students are very enthusiastic about core agile practices and that there are no significant differences in the perceptions of students of various levels of educational programs. Overall, the results indicate that a broad range of students (although not everyone) accepts and likes agile practices. This holds for all ages and for various degrees of prior industry experience. Qualitative insights reveal that experience of working as an agile team promotes the development of professional skills (communication, commitment, cooperation, and adaptability). These positive views are, in our opinion, a prerequisite for the widespread adoption of agile methods in industry.

The overwhelmingly positive perceptions on agile methods seem to indicate that developers too will support the adoption of agile practices wholeheartedly (beyond merely paying lip service to the methods because they are forced onto them by management). Getting the buy in of front line workers for any disciplined process increases the chances to reap real benefits. Thus, while quantitative, industrial evidence of the efficiency and benefits of agile practices is still sparsely available (or, for some practices, not available at all), the results of the present study lead us to believe that agile approaches will have a substantial impact on software teams. Our future work will focus on validating this hope. Leaders of agile software teams interested in collaborating should contact the first author.

7. ACKNOWLEDGMENTS

The authors would like to thank all students from SAIT and University of Calgary who provided us with their thoughtful responses. The study is partially sponsored by NSERC, iCore, and ASERC.

8. REFERENCES

[1] Bedoll, R. A Tail of Two Projects: How 'Agile' Methods Succeeded after 'Traditional' Methods Had Failed in a Critical System-Development Project. *Proc. XP/Agile Universe 2003, LNCS 2753*: 25–34, 2003.

[2] Boehm, B. Get Ready for Agile Methods, with Care. *IEEE Computer*, 35(1): 64–69, 2002.

[3] Boehm, B. and Turner, R. *Balancing Agility and Discipline: A Guide for the Perplexed*. Addison-Wesley, Reading, MA, 2003.

[4] Ciolkowski, M., Muthig, D., and Rech, J. Using Academic Courses for Empirical Validation of Software Development Processes. *Proc. EUROMICRO/ SPPI04*, IEEE Computer Society: 354–361, 2004.

[5] Highsmith, J. and Cockburn, A. Agile Software Development: The Business of Innovation. *IEEE Computer,* 34(9): 120–127, 2001.

[6] Höst, M., Regnell, B., and Wohlin, C. Using Students as Subjects – A Comparative Study of Students and Professionals in Lead-Time Impact Assessment. *Journal of Empirical Software Engineering,* 5(3): 201–214, 2000.

[7] Humphrey, W. Comments on eXtreme Programming. *IEEE Computer Society Dynabook*. Online http://computer.org/software/dynabook/HumphreyCom.htm Last accessed on August 30, 2004.

[8] IEEE Computer Society/ACM Joint Taskforce on Computing Curricula. *Computing Curriculum- Software Engineering* (public draft 1), July 17, 2003.

[9] Melnik, G. and Maurer, F. Introducing Agile Methods in Learning Environments: Lessons Learnt. *Proc. XP/Agile Universe 2003, LNCS 2753,* Springer Verlag,: 172–184, 2003.

[10] Melnik, G. and Maurer, F. Introducing Agile Methods: Three Years of Experience. *Proc. EUROMICRO/ SPPI04*, IEEE Computer Society: 334–341, 2004.

[11] Melnik, G., Read, K., and Maurer, F. Suitability of FIT User Acceptance Tests for Specifying Functional Requirements: Developer Perspective. *Proc. XP/Agile Universe 2004*, *LNCS 3134*, Springer Verlag, 2004.

[12] Stephens, M. and Rosenberg, D. *Extreme Programming Refactored: The Case Against XP*. APress, Berkley, CA, 2003.