Student Experience of Using Agile Development Methods in Industrial Experience Projects

Roy Morien
Research Fellow
roymorien@hotmail.com
Centre for Extended Enterprise and Business Intelligence
Curtin Business School
Curtin University of Technology, Perth, Australia

ABSTRACT

The paper will report upon the experiences and reflections of undergraduate students undertaking industrial experience projects in their IT/IS courses. The students utilized an agile, adaptive development approach, sometimes known as a lightweight methodology, or an evolutionary, incremental approach. This paper is discusses the experience of the students, as stated in a substantial feedback questionnaire, and weekly diaries.

Keywords: Agile Development Methods, Evolutionary Development Method, Incremental, Evolutionary Development Method Focal Entity Prototyping, Lightweight Methodologies, Student Industrial Projects.

1. INTRODUCTION

During 2003 and 2004 some 220 students, in 57 groups, in the Bachelor of Commerce courses in Information Systems, Information Technology and Electronic Commerce (and combinations) in the School of Information Systems at Curtin University of Technology undertook a major industry-based system development project in their final year of study. The student project groups were requested to undertake their projects in an "agile and adaptive" manner; incremental development, prototyping approach, early commencement of construction, highly client orientated. Rigorous attention to minimal documentation was emphasized (that is, essential to communicate necessary information to future stakeholders, but no more).

The attitudes expressed towards these development methods, the acceptance and adoption, or otherwise, of the methods, and the personal experience, attitudinal and system outcomes of the projects was monitored, partly by regular, weekly or fortnightly "reflective" feedback documents, and a significant questionnaire at the end of the projects.

These comments, reflections and responses, together with personal observations and discussions with students, were analyzed and collated.

2. PREFERENCE FOR TRADITIONAL METHODS IN CURRICULUM

A distinguishing factor in the student industrial experience projects for 2003 and 2004 was that in these two years agile development methods were emphasized for the first time. Prior, subsequent and parallel cohorts of students have usually been required to follow a traditional Waterfall Model / Structured Design Life Cycle (SDLC) approach.

These other cohorts of project students had been given the opportunity to select their system development approach, but this has been somewhat of a vain option. Teaching of Prototyping, as an example, has been done in one, possibly two lectures in a previous Analysis unit, and has been at least strongly implied as being a minor option to the more acceptable Waterfall Model / SDLC based approach. In reality, the students had been given so little information about the alternatives to the Waterfall Approach / SDLC based approach that it was almost inevitable that they chose that option only.

Of greater importance was the fact that it was considered that the students had never been given any realistic information or assumptions that the agile/ rapid /prototyping approaches were in fact acceptable in industry. Students, frankly, just did not believe, or know, that it is a valid and useful option to develop in a 'lightweight' manner. A heavily document oriented approach has always been emphasized as the right thing to do. In fact, the projects were, and still are, divided into 2 separate units. These are known as ISP391 and ISP392, and run sequentially. The first unit has always been seen as the "Analysis & Design" phase, where a Requirements document was the only required outcome, with the construction activity restricted to the second semester. This assumption is institutionalized in the unit curriculum statements, and in information given to students at the commencement of the project. This was also supported in a document entitled "Project Frequently Asked Questions" (FAQ's), given to the previous cohorts of students at the commencement of the first project unit that poses, as the very first question: "**Question:** Our client wants a prototype in 3 months. What do we say? **Answer:** Say NO! One of the outcomes of this unit is to produce an Analysis and Design document which will be used in the following unit ISP392 to build the system".

These emphases on Analysis & Design only, and the predicated outcome of documentation only, does not, it is suggested, provide appropriate guidelines to the students on what they must do and how they are to go about their task. Firstly, the students are confronted with a rather large and amorphous task of "doing analysis and design", for a whole semester. Frankly, this leaves many students confused about exactly what they are supposed to achieve, and so they set their sights on producing the largest and most impressive looking document that they can; the document is seen as the important outcome, as a proxy for truly gaining domain knowledge and proposing and agreeing upon design options and outcomes. This leads to the inevitable next questions; is the document useful, or useable? Is the content of the document valid? Does the document actually provide an appropriate blueprint for the subsequent development? Unfortunately, the answer to these questions is frequently No! Merely having a beautiful Data Flow Diagram (DFD) does not in any way imply that it is valid and indicates reality, or agreement with the client. Frequently the Entity Relationship Diagram (ER Diagram), apart from being usually confused with a Relational Data Diagram, has been drawn up strictly according to ER Diagramming Rules, and does not obviously support the business processes or information architecture. So much time and effort goes into producing "professional" documentation that it becomes a document producing exercise, not a true "requirements elicitation, analysis and design" activity. Unfortunately, this time consuming exercise still cannot in any way guarantee that the content of the documentation is valid, reliable, accurate or even realistic. It was common to make substantial changes to the Requirements Documents in the next semester, when construction was undertaken. In fact, there was a rigorous process in place, requiring the full documentation of any changes that were made, including changes to the documentation.

3. ATTITUDES TOWARDS AGILE DEVELOPMENT

This was one very interesting and valuable lesson learned by the students; that adopting an agile approach actually gave the students a very well structured project management approach, and they were able to start producing positive outcomes almost immediately. Far from being the ad hoc, "quick and dirty" approach that it is often characterized as, agile development, using an agile development method called SCRUM in this case, proved very well organized, very structured and kept the projects very much on track right from the very beginning (after the students had come to terms with it, that is).

(Details of SCRUM can be found in Schwaber & Beedle M. (2001), and at http://www.controlchaos.com).

The more "social" aspect of this approach is that this documentation-only policy actually becomes very tedious and boring for the students, who then do not do it well.

One very unfortunate observation can be made here, and that is that it appears that most Universities, world-wide, prefer to include the late-1970s Waterfall / SDLC based approaches, and may even deprecate the agile / lightweight methods if they are mentioned at all. This presumption of University preference in teaching is supported by the fact that recent research (IBM, 1999) has shown that DFDs are the most popular tool taught in systems analysis and design courses: 597 out of 647 schools (92 percent)

indicated that they teach DFDs in that course". DFDs are seen as essentially being part of a Waterfall / SDLC based approaches. This, combined with the lack of apparent acceptability of the agile methods, had a significant affect in constraining the students from adopting these "alternative" methods. In the face of the dismal history of system development failure, presumably using the SDLC approaches, it remains an unanswered question as to why Universities continue to teach these development approaches as if they are the received way to do that.

It was this attitudinal predicament that needed to be overcome, as well as attempting to ensure that the students had significant exposure to both development approaches – what may be categorized as Heavyweight Methods versus Lightweight Methods. As students about to graduate, they had very little if any realistic experience in system development, beyond their usual development assignments. Given this, it was felt that perhaps they would be able to be influenced at this early stage of their professional careers by a good experience to see these new methods as being useful, and acceptable.

4. TEACHING STUDENTS AND LEARNING BY DOING

Given the predicament of most if not all of the students having little knowledge of agile approaches, it was incumbent upon the subject lecturer to provide students with some knowledge, at least to get them started. This actually was not a difficult task, because of the relative simplicity of the agile approaches. The most difficult part of this was to overcome the already embedded assumptions in the students' minds that significant up-front documentation was what was important. The agile approaches emphasize early and frequent delivery of working code. A constant value stream is required.

In fact, using the "Focal Entity Prototyping Approach", as espoused by the lecturer, it is possible to have sufficient information from the client to commence development immediately after the first meetings with the client take place. As with many aspects of the agile approach, and the agile project management approach, this thinking runs quite counter-intuitive to the accepted wisdoms of the traditional school of thought.

Therefore, the major problem, at the start, was not imparting the new information to the students about the agile approaches. The main problem was overcoming the students' view that it was difficult because it was different, and they had previously been told something entirely different.

The textbook that the students were required to purchase, and to be regularly quizzed upon at weekly supervisor meetings, was "Lean Software Development: An Agile Toolkit" (Poppendieck & Poppendieck, 2003) to formalize the adoption of an agile approach. This is a major book on the nexus between lean product development principles and agile software development. Other books of great merit on the subject of lean product development include those by Womack and Jones (2003; Womack, Jones & Roos (1991), Kennedy (2003), Liker (2004) and Gross & McInnis (2003).

During the first half of the project period, about 12 weeks, it was clear that the students were struggling with the concepts and practices, but were gaining experience and expertise, as well as becoming much more accepting of the agile approaches and lean thinking. It can safely be said that by the end of the project, most of the students had become enthusiastic about using these approaches. Going purely on their experience, many had started to deprecate the traditional, rigorous, phased approaches.

5. THE AGILE MANIFESTO

Before continuing, it is useful to describe exactly what it is that is meant by an agile approach. The best source for this understanding is the Agile Manifesto, published by the Agile Consortium. The major points of this manifesto are; through this work, we have come to value:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

Some of the principles underlying the Agile Manifesto include:

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Working software is the primary measure of progress.
- Continuous attention to technical excellence and good design enhances agility.

From these principles, and the manifesto itself, it can be seen that agile development approaches are not merely iterative approaches. They are based on a clear set of philosophical considerations. There is also no doubt that much of the thinking in regard to agile development methods can be traced back to both the experience of a group of highly experienced developers, as well as the principles of lean product development.

6. AFTER GRADUATION – EMPLOYER VIEWS

A number of students reported back that, in their job interviews with prospective employers following their graduation, the area of the undergraduate studies that those employers focused on in the main was the industrial experience projects. Each of those students indicated that the employer was impressed by the fact that the students had not only some good knowledge of the agile / lightweight development methods generally, but also had hands-on experience of applying such methods. In at least one case the student acknowledged this as the main reason that he was offered the position. To quote from an email from that student "In my various interviews in my job search, I can honestly say that the local companies I talked to (at least 10 different companies), not one of them used the traditional approach at all. In fact, they were surprised that the waterfall approach is still taught at all. In contrast, they were very impressed that I knew quite a bit about Agile Development and even had practical experience of using it. Moreover, I even got a job! This position involves the usage of agile methodologies."

7. DEVELOPMENT VISIBILITY AND CLIENT INVOLVEMENT

By adopting an agile, incremental, iterative approach, where actual working components of the system were delivered on a regular, usually weekly, timetable, progress on the development of the system was highly visible, and delivered outcomes can be reviewed by the Client, or by the Project Manager. Every completed task, be it a documentation task, or a programmed task, or a research task, can be scrutinized for correctness and completeness i.e. validated. The project manager is able to ascertain the productivity, competence and quality of the individual project group members, and appropriate action taken to rectify or overcome any difficulties seen to be arising, at a very early stage of identification.

In the student project situation, this approach was seen to be highly beneficial, because an on-going regime of time management could be enforced, avoiding the "last minute panic" approach to task completion as the end of semester approached. Where there were deliverables on a weekly basis, contractually undertaken by the students, these can be monitored on a consistent and constant basis over the whole semester.

Although these attributes of the approach are seen as being especially useful in guiding students' work and progress, they are just as useful and relevant in an industrial project manned by competent and experienced professional developers. Thus it is a highly effective and efficient approach to the project lifecycle.

8. DATA COLLECTION

On a regular, usually weekly basis, all students were required to present certain documentation to their project supervisor.

Group Level Documentation & Feedback

The group as a whole was required to present

- Project Folder, with updated memos, notes, progress reports,
- Project Backlog List stating all known tasks to be completed, with a time estimate, and a prioritization such that the tasks with the highest priority were the next to be allocated.
- Project Update Report.
- Data Dictionary, especially showing updates since "last time"
  - Entity Definition Sheets
  - Relationship Definition Sheets,
  - Attribute Definition Sheets,
  - Table Definition Sheets
  - Data Field Definition Sheets
  - (all according to pre-stated templates)
- Working System Prototype, demonstrating progress so far.

Individual Developer Documentation & Feedback

Each individual student was required to present

- Personal Sprint Backlog List (for the forthcoming sprint – a SCRUM term for an iteration period),
- Personal Timesheet, fully classified and dissected into development activity categories,
- Personal Reflective Log
- Constructed outcomes, referenced on the Timesheet.

By this means an information continuum for project management (and assessment) purposes was clearly established. The Personal Sprint Backlog List was a statement of Planned Activity, the Timesheet was a statement of Actual Activity (able to be tied back to the Plan), and the delivered and constructed outcomes that were able to be demonstrated could be traced back to the timesheet, that referenced these outcomes. So a Plan – Act – Reflect cycle was established.

Some very interesting outcomes of this, especially in the keeping of the Personal Timesheets, were that the students were constantly surprised at how inaccurate their initial estimates were, how time-consuming and therefore expensive the development activity actually was, and how difficult it was to apparently correctly classify their activities, especially the classification of analysis, design and construction activities. For example, sitting with a client and using a report generator to create a report layout cannot be classified directly into Design, or into Construction. If the report creating session was also giving the client the opportunity to suggest new and different reports, that was difficult to classify simply under Analysis.

End of Project Questionnaire

At the end of the project, a substantial questionnaire was completed by each group, and used as the basis and structure of a rigorous examination, by way of presentation and defense of their outcomes by the students under questioning from myself, as Unit Leader.

By these various means, a substantial amount of information about the students' activities, personal views, reflections, opinions, complaints and achievements was elicited. The information was collected in a variety of ways, over an extended period of time; each week for most of the 24-week period of the project. It is felt that the feedback was not contrived for "assessment purposes only", and did accurately and appropriately reflect the students' views over time. Being essentially a longitudinal study, it reflected the changing views as the students became more experienced in the project activity and all its methodological and technological aspects.

9. STUDENT VIEWS ON THE AGILE METHODS

Question: Discuss and comment upon your experience of how efficient and effective the use of an evolutionary / prototyping approach to your project was, or could have been.

It can be confidently stated that almost without exception the students felt that these development approaches were indeed efficient and effective. The breaking down of the project into discrete sprints, once understood and accepted, proved to be a valuable innovation, and allowed the students to carefully step through their project in an orderly manner.

Student comments under this question included:

"efficient by completing useful 'chunks' of the system and then moving on"

"it enabled us to more quickly determine requirements"

"early learning was helpful in later iterations"

As indicated elsewhere, those groups that attempted to initially follow a more phased approach quickly found them struggling with detail, but once they adopted this iterative, agile model, they found that progress was good, orderly and efficient.

10. IMPACT ON THE GROUP, MORALE, COMPETENCE, INTEREST

There is a perception amongst many students that systems development is rather boring and tedious. This is not a researched attitude, but is supported anecdotally. The author, as Unit Leader, and as a committed systems development professional of nearly 30 years' experience, more than 20 of those years in an academic position, was very keen to dispel this attitude, and to try to demonstrate to the students that systems development is an interesting, possibly exciting, but certainly dynamic career to enter.

System development is seen to be very much a social activity, as well as a technological activity. People work together in groups, which is social. Developers work together with Clients, which is social.

The author was keen to see how students felt about this situation.

Table 1: Analysis of Question: Do you think that presenting completed and working increments on a regular basis added to your group's confidence and positive attitude during the project?

• YES, I believe this to be the case	57
• NO, I don't think it made any difference at all.	0
• Can't Say – I don't know how we would have felt otherwise.	0

That is, every single project group indicated that their interest, their confidence, their positive attitude was enhanced or maintained by this highly iterative, incremental delivery, approach. They indicated that at every step there was something to see, something that worked, and that they could demonstrate to the client. They learned at every step, they gained confidence at every step.

An excellent outcome.

11. ADOPTION OF STANDARDS, AND A REUSE POLICY

To achieve the productivity necessary when adopting an agile development approach, reuse of development components and artifacts is an essential part of the development tactics. Students were asked to discuss the policies and practices that the group stated, and practiced, about the reuse of system and programming components and artifacts. Their responses were, in general, very positive.

Reusable artifacts included documentation templates, standard form layouts (look & feel standards), standard code, naming standards, coding standards. Students obviously saw the benefits of not having to invent their own documentation layouts etc. Comments in this regard include

"We strongly agree that it was effective and useful …"

"It is extremely efficient…"

"It is quick to adopt, and really saved us time"

"Reuse policies saved the group quite a large amount of time … we realized how much time had gone into the development of (the given) standard forms…. We quickly realized that standardization was important for any system, and had learnt that in future trying to stick with on type of standardized forms can and will cut down a great deal on the time to develop a system" (OZMAU Group)

This last comment was especially interesting as the students were acknowledging that they now understood the effort necessary to create an "easy to use" development environment, and the payoff from that in subsequent system development.

Some specific outcomes of the adoption of a policy of standards, standardization and reuse, were elicited.

Table 2: Analysis of Question: Having well-stated standards was beneficial to the good order and efficiency of your system development activity?

It was interesting to see the conflicts and disagreements, and the amount of rework undertaken, by groups who did not start out with clear ideas about their standards, and their intention to reuse. There was indeed a lot of conflict, and a lot of time wasted, while groups were sorting these matters out.

1 means "Not at All", 5 means "Excellent" 2,3,4 are varying degrees of success. ➔	1	2	3	4	5
• Harmony between group members	1	4	8	11	22
• Productivity			5	18	20
• Quality Outcomes			6	22	26
• Acceptability by the Client			6	11	27
• Enhancing member's knowledge		1	8	14	17
• Other (Time taken to develop some functions)					1

However, most students came to very satisfactory realizations about this, and the figures in Table 2 clearly indicate that standards and reuse variously enhance group harmony, productivity and quality to a significant extent.

12. INTERACTION WITH AND VIEWS ABOUT CLIENTS

When students are taught the waterfall / structured analysis approach to systems development, which seems to be the most usual development approach written about in textbooks, and taught in University courses today, there is an implication that "all will be well". That is, your task is to elicit requirements from clients, and clients will be there and able to had requirements elicited from them. The underlying implication of the fully structured waterfall approach is the statement that "Once the client has indicated their requirements fully and in great detail, right up front, then we will be able to deliver the optimal system to satisfy those requirements". Unfortunately, this has proven to be invalid in practice. The history and record of system failures over the last 30 years, and continuing, denies this simplistic assumption.

So what did the students find?

Question: Do you think that it is possible or feasible for your client to state all of the requirements for the system in the first phase of your development activity?

All but two groups said No, not possible or feasible. However, both of the client organizations for the two standout groups were technology companies, and the personnel involved were experienced system developers who did know what it was that they wanted. Student comments included:

> *"The client cannot envisage the final product"*
>
> *"It would be nice to receive all this information, but doesn't seem possible."*
>
> *"As (the client) got to see the progress of the prototype … she came up with new ideas/functionalities as to what she wanted the system to do."*
>
> *"We don't believe that this was possible as the client didn't appear to understand the current system or all the requirements correctly"*
>
> *"No. it is not possible because more and more requirement will be added from time to time when we started to show them our increments".* (Red Rossini Group)

These comments indicate the "learning" outcomes of an evolutionary approach, whereby both clients and developers had continuing opportunities to learn about requirements, about possibilities and options, at a time when those realizations can be incorporated in the system. In addition, as the project proceeds, both client and developer improve at knowing, understanding, and stating requirements.

The groups cited so far were all interacting with small, unsophisticated clients, as indeed most of the project groups were. There was one group, however, who were embedded in a large organization, with a large IT department.

At a group presentation to this latter client, an interesting contradiction arose that was commented on and discussed, at one of these meetings. First, it was evident that the client representatives, about 5 in all, were all eager to suggest modifications, and to discuss shortcomings and suggest solutions. As this client group changed from meeting to meeting (one person missing and replaced by another, was typical) then new client representatives were often present. They were willing and eager to suggest changes, enhancements, and to point out shortcomings. However, at the same time these experienced project managers and IT practitioners often used the term "scope creep" in a somber and portentous manner. This obvious contradiction was pointed out, and the client representatives realized immediately the contradiction between their enthusiasms for evolutionary or incremental changes, and the fact that this was "scope creep" that they were so keen to avoid. It was demonstrated to them that changes were valuably incorporated, and could be incorporated, given the agile development approach being undertaken, without the project becoming a runaway example of uncontrolled "scope creep".

Question: Did your client ask for new or additional features and capabilities during the course of the development.

Every group answered yes to this question. Every client asked for more, or different. There was a very clear demonstration of the "learning" affect of the evolutionary development approaches, and the benefits of providing clients with working prototypes on a regular basis.

Question: Do you think that showing your client completed features of the system, and installing a working prototype at regular intervals, was useful in

reaching agreement with your client, eliciting necessary features etc.

Every group answered yes to this question. Comments from students included:

> *"Yes, with demonstration of prototypes on regular basis, client can see the improvement and had a better understanding as well as to check whether the new prototype fulfill their requirements"* (Infotech Group)
>
> *"Yes, the reason for us answering yes is because we believe by showing the client our system and the added functionality at regular interval, we can gain the client's agreement on the functionality and the design which we had shown him at that particular point in time. By having his agreement on this matter, we can be sure that he is happy with everything and it is all right for us to move on to the next step of development. Hence, our project team doesn't have to go back and forward trying to change things that we had already finished."* (Tech 5 Group)
>
> *"Yes. By presenting our prototype to the client regularly not only can we gain the trust from client but also can get feedback from them before we move to the next step.* (IT Consultants Group)

Table 3: Analysis of Question: Do you think that presenting the prototype to your client early in the project gave your client more confidence that you are able to do the job?

• YES, I believe this to be the case	26
• NO, the Client wasn't impressed	3
• Can't Say – the Client has not indicated anything about this	7

13. PROJECT OUTCOMES – THE BOTTOM LINE

Of the 57 groups, four groups failed but were given a supplementary activity to complete, and one group failed totally. The other 52 groups passed with varying degrees of success.

At the final two-hour presentation session, clients were invited to be present, and their comments and feedback elicited. Clients were also requested to provide written, confidential feedback.

Overall, it can be said that the groups showed greater interest, created better systems that were more complete, and the clients were generally extremely happy with the outcomes.

Of course there were problems that arose because clients could not be available on a regular and consistent basis. Unfortunately a number of the clients took the view that "it is only a student project" and did not take the situation very seriously. However, some systems were developed for major clients in sizeable organizations, and there were many stories of highly enthusiastic acceptance of the project outcomes.

It was just not possible to have a situation that allowed a comparison between outcomes achieved by developing the systems using both traditional and lightweight approaches. However, at the very least it can be said that the use of agile approaches to the development was highly successful, and the students identified from their own personal experience the inherent difficulties present in systems development, and the ability to overcome these difficulties using agile approaches.

Pedagogically, it was clearly a significant and successful outcome, for the students to learn about and experience the use of agile approaches to systems development.

14. CONCLUSION

At least at this University, the traditional development approaches were favored, vigorously taught, and only taught. Agile development was deprecated and all but ignored.

It was clear at the beginning of the project that students were, in general, quite bewildered at what they were being asked to do. This was, to a great degree, attributable to the fact that most if not all of the students had received little prior information about what may be called the new contemporary lightweight methodologies (although "new" is hardly accurate, with prototyping literature discussing this back in the early 1980's, and iterative development being discussed back into the 1950's). There was a definite "This can't be right?" attitude amongst many students.

Nonetheless, when the students were required to undertake industry experience projects, they gained significant knowledge and experience from this experience. Nearly every one of 220 students indicated this. They went from an attitude of doubt, or even suspicion, and sometimes resentment, because this was different to what they had learned before, to an attitude of positive and

enthusiastic embracing of this approach. It must be said that, as with so many things, each student had their learning experience enhanced by as much as they were willing to embrace the opportunity.

In over 55 projects it was demonstrated that most clients either do not fully understand their own requirements, or cannot state those requirements up front, or are quite willing to propose new requirements as the project proceeds.

The highly visible nature of the prototyping approach had the effect of empowering the students and giving them greater confidence in their own abilities. It also provides the opportunities necessary for learning to take place, and for the system actual deliverables to converge on the system required deliverables at delivery stage. Although it may seem to provide substantial project management problems, and problems of "scope creep", the result is in fact substantially beneficial, in that the client gets a useful and useable system, and the expectation gap between what is really required, and what is delivered, is narrowed substantially, or eradicated completely. Significant project "overrun" was not seen as an inevitable outcome, and indeed was not seen as a problem at all.

Project management is not invalidated or put at risk, as many managers may fear. The use of the highly iterative "sprint", the visibility of the Project Backlog List, with estimated and prioritized requirements stated, makes project management, if anything, simpler, and makes estimates more relevant, accurate and realistic. Every requested change can still be scrutinized and estimated, but can be immediately shown to impact the project in some way, and its priority and essentiality can be identified.

Overall, most of the student groups felt that the adoption of agile development approaches to systems development was appropriately structured, manageable, and controllable and provided an orderly approach that was rapid and effective.

Viewing this situation as a research project, the hypothesis can be stated, "The adoption of Agile Development Methods for system development is appropriate and leads to project success". With feedback from 57 groups, encompassing the views, opinions and experience of 220 students, it can be at least tentatively stated that this hypothesis has been proven.

The evidence is clear. Nearly unencumbered by the baggage of experience, these students have given a resounding yes to these new, contemporary approaches. They have given a distinct NO to the traditional rigorous pre-definitional development approaches, as manifested in the "Waterfall / SDLC" approaches. These latter approaches still seem to be the mainstay of conventional systems development in most organizations today, and are, as well, the major subject matter of most systems analysis and design courses in Universities.

The students directly experienced the realities of system development, and extrapolated from that the preference for the agile approach to development.

At the very least it is incumbent on colleges and universities to now include the agile / adaptive development approaches in their Information Systems curricula, and also agile project management, which has not been discussed here, but must be implied as necessary, as a competent way to manage agile projects.

References

IBM (199), *IBM Research Journal*, http://www.research.ibm.com/journal/sj/381/millet.html, accessed December 13th, 2003.

Kennedy, Michael N. (2003) Product Development for the Lean Enterprise, The Oakley Press.

Liker, Jeffrey K. (2004) The Toyota Way, McGraw-Hill

Poppendieck M. (2003), Home Page, http://www.poppendieck.com/, accessed December 14th, 2003.

Poppendieck M. & Poppendieck T. (2003). Lean Software Development: An Agile Toolkit, Addison-Wesley, Pearson Education.

Schwaber, K. & Beedle M. (2001) *Agile Development with Scrum*, Prentice-Hall, 1st Ed.

Womack, James P., Daniel T. Jones & Daniel Roos (1991) The Machine That Changed The World, HarperPerennial.

Womack, James P. & Daniel T. Jones (2003) Lean Thinking, FreePress.