Insights into Using Agile Development Methods in Student Final Year Projects

Roy I. Morien
Curtin University of Technology, Perth, WA, Australia

morienr@cbs.curtin.edu.au

Abstract

The paper will report upon the experiences and reflections of undergraduate students undertaking industrial development projects, as part of their curriculum, in the situation where they have been required to utilize a prototyping / agile / lightweight methodology. The paper is partly evangelistic (demonstrating the author's enthusiasm for a particular development methodological viewpoint), and partly methodological (discussing the characteristics of, and success in using, a particular methodological approach). A clear position in support of agile development methods is taken, and a particular method, called the Focal Entity Prototyping Approach, published by the author, is a main item of interest in this discussion.

Keywords: Agile Development Methods, System Prototyping, Focal Entity Prototyping, Lightweight Methodologies, Student Industrial Projects.

Introduction

Students 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 are required to undertake a major industry-based system development project in their final year of study. In 2003, the project units were lead by an academic who has a long history of encouraging the use of what are now called variously "Lightweight Methods", or "Agile Methods" of systems development (Fowler, 2000, 2003; Svetinovic & Godfrey, 2003). The student project groups were strongly recommended to undertake their projects in such a manner; incremental development, prototyping approach, early commencement of construction. Rigorous attention to minimal documentation was emphasized (that is, essential to communicate necessary information to future stakeholders, but no more).

Thirty-five student groups, comprising 135 final year students usually in groups of four, commenced an industrial project development activity in March 2003. This project activity is divided into two units, referred to here as ISP391 and ISP392 that run in successive semesters. Contrary to the manner in which prior running of the units was undertaken, the students were specifically required to adopt a different approach to system development, and the project management method previously taught.

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 have been analyzed and collated, together with personal observations and discussions with students.

Lightweight Development Methods in Student Projects

What we may generally call "lightweight" methods (Fowler, 2003) can be seen to encompass a variety of named approaches and methods. SCRUM (Agile Alliance, 2003; Schwaber & Beedle, 2001), CRYSTAL Methods (Cockburn, 2003), DSDM (DSDM Consortium, 2003), Lean Development (Poppendieck, 2003; Poppendieck & Poppendieck 2003) and Extreme Programming (Extremeprogramming.org, 2003) are all varieties of "agile" development methods, which may also be seen to encompass Rapid Application Development, Rapid Development (McConnell, 1996), and more general terms such as iterative development, incremental development, prototyping (Bemelmans, 1983; Budde, Kuhlenkamp et al, 1984; Burns & Dennis, 1985; Gilhooley, 1986; Hawgood, 1981; Naumann & Jenkins, 1982; Sumner, 1985; Vacca, 1984). All have the same underlying principles of high visibility development, client oriented, iterative, incremental and adaptive.

Other cohorts of project students have 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 minority option to the more acceptable SDLC / Waterfall Approach. Frankly, the students have been given so little information about the alternatives to the SDLC / Waterfall 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 industrially acceptable. The immediate acceptance of the view that "It is OK to develop in a 'lightweight' manner" was just not prevalent. A heavily document oriented approach has always been emphasized as the right thing to do. In fact, the projects are divided into 2 separate units that run sequentially, and the first unit has always been seen as the "Analysis & Design" phase, 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. The curriculum statement for the first project unit states "Analysis …. Design …. Project Management …" but does not mention any word about Construction. The subsequent unit curriculum does include Program Design and Construction. This division of activities between the two units is supported in a document entitled "Project Frequently Asked Questions", 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 in our view, 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 does not in any way imply that it is valid and indicates reality, or agreement with the client. Frequently the Entity Relationship 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.

This view is supported by comments of one group of students who said, "…we did not possess the business analysis acumen to enable us to fully comprehend a set of system requirements, which are in a state of flux generally. The problems that we encountered during prototyping may have become insurmountable if we had adopted the Waterfall Approach." (EyeQ Group Project Questionnaire).

These are the two major problems seen as arising. These are lack of process knowledge by the students, and lack of guarantees of validity of their voluminous documentation.

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.

Much of the "agile / lightweight" literature emphasizes practices that may be seen to run counter-intuitive to the conventional wisdoms of the SDLC / Waterfall Approaches that are usually taught in Universities. 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 SDLC / Waterfall Approach. 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 the lecturer was attempting to 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, albeit about to graduate, they have 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.

SCRUM as a Predicated Approach

The students were instructed to adopt a specific agile approach (or at least major elements of it), and the indicated approach is called SCRUM. This development method is proposed and supported by the Agile Alliance (Agile Alliance, 2003; Schwaber & Beedle, 2001). The primary characteristics of the SCRUM approach are:

A time-boxed, iterative development approach (a Sprint),

The project activity is undertaken primarily through a series of short period incremental cycles, called "sprints". The time box may be a week, or 2 weeks, or up to 3 months, but usually less than a month.

Maintenance of a Project Backlog,

There is an overall Project Backlog of Requirements maintained, that is updated regularly as further requirements are discovered, and known requirements are satisfied and delivered. Each requirement on the backlog list is prioritized and estimated, so that, within appropriate allowances

for uncertainty, the project schedule and budget requirements are known. New requirements are more easily identified, and are prioritized and estimated as they are placed on the backlog list.

Maintenance of Personal Sprint Backlogs,

At the commencement of each sprint, each member of the project team moves some requirements from the Project Backlog List to their Personal Sprint Backlog. The delivery of those requirements is the sole focus of that project group member, for that sprint. This is often seen as a contract with the group member, who is able to allocate themselves only the amount of work that can be successfully completed in that sprint. Within reasonable bounds, the project group member cannot and will not accept any changes to their sprint activity, as undertaken at the start of the sprint. Any new project requirements that may arise are immediately directed to the Project Manager, away from the group member, and placed on the Project Backlog List. The individual group member thereby moves towards a successful completion of the designated tasks without interference or deflection from the task.

This "fixed requirements" situation is not seen as a problem, because of the short timeframe being considered. Nothing is seen as being so urgent or imperative that it cannot wait to be attended to for maybe 2 weeks or less.

Project meetings regime

Each sprint is commenced with a meeting to allocate (or request) tasks for each developer. Items are transferred from the project Backlog List to individual personal Sprint Backlog Lists. On a daily basis a brief "standup" meeting is held to allow group members to communicate with each other about their accomplishments for the day, and any problems that may have arisen. This daily meeting adds to the cohesion of the group by ensuring that all group members know enough about the overall activity to be able to continue their own activities in an efficient and informed manner.

At the end of each sprint there is a meeting to sign off the outcomes of the sprint. There is a clear intention that selected task can be seen as completed, increments to the system can be delivered, and items on the Project Backlog List can be signed off as complete and delivered.

Development Visibility and Client Involvement

By these means, progress on the development of the system is 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 ongoing 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.

The Focal Entity Prototyping Approach

Having scrutinized the literature on prototyping, as long ago as 1990, and subsequently, it became obvious to this author that most or all research papers and articles on the subject of system prototyping constantly referred to "the system" being prototyped. There did not appear to be any clear indication of what this author has termed "a prototypable unit of work". Put simply, if prototyping was a highly visual and iterative approach, what was it that was being visualized and constructed in any given iteration? This absence of focus perceived in the literature seems to be supported in Berghel (1996) where he refers to "vacuous prototypes".

One answer that came to mind seemed to be simply "an entity". The Focal Entity Prototyping Approach is centered on the identification of an Entity, and the subsequent and ultimate delivery of analysis, design and programming artifacts and outcomes based on that entity (Morien, 1993). This realization was possibly prompted by such as Appleton (1983).

To illustrate, two models of development are proposed; one generally termed a strategic model and the other termed a tactical model (see Figures 1 & 2).

The "Focal Entity Prototyping" Strategic Model

There are **iterative feedback cycles** implied within this model:

Step 3 → Step 1: A new entity is discovered that must be implemented, and activities return to Step 1 to incorporate that new entity in the Working Entity Model.

Step 4 → Step 3: This is the fundamental prototyping feedback cycle, of development / user evaluation / modifications requested / development.

Step 5 → Step 1 or Step 2: On successful completion of the entity-based increment, return

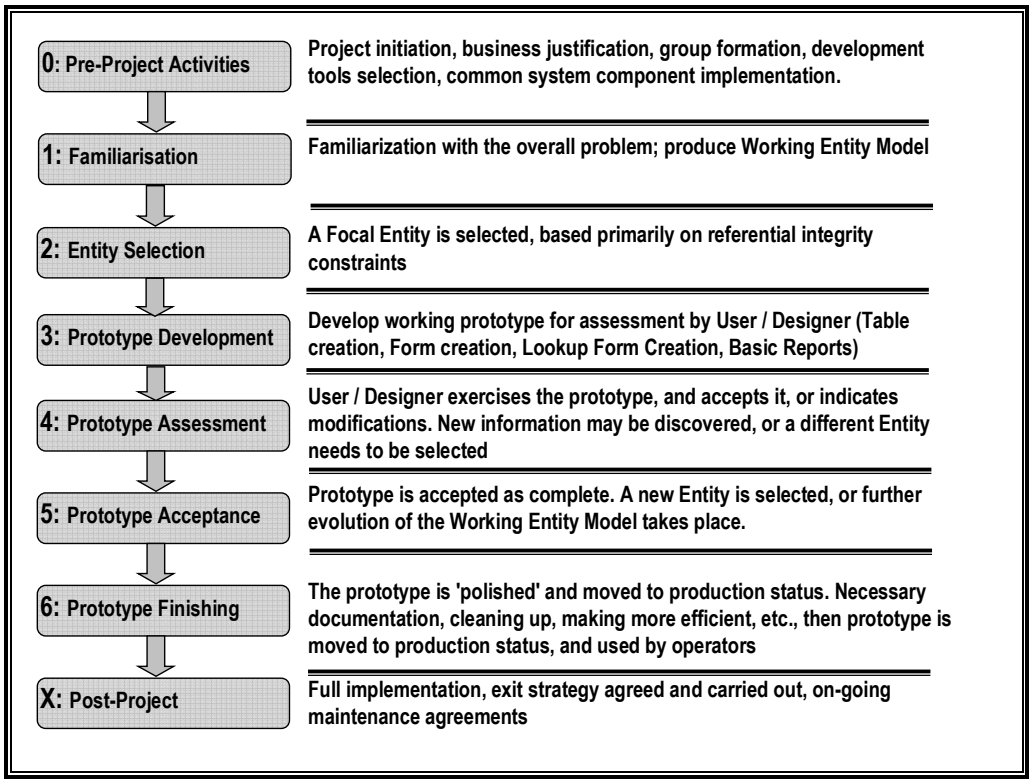

Figure 1: The Focal Entity Prototyping Strategic Model (Source: Morien, 1994)

whether to select a new entity, or back to Familiarization to extend the Working Entity Model by the discovery of new entities or relationships.

The Focal Entity Approach Tactical Model

It is the combination of Steps 2 and 3 that are elaborated in the Tactical Model of Development (Figure 2).

It must be stated here that this author has the view that Entity Relationship Modeling is much more than just a database structuring technique. In our view, ER Modeling realistically and appropriately encompasses the processes that are implied by the existence of an entity and/or a relationship. This view is not unsupported in the literature, from a wide variety of other authors. Simsion (1994, p19) for example states "In defining the data we intend to store, we have implicitly, and very concisely, identified a whole set of functions to capture, display, update and delete the data." Elmasri & Navathe (2000, p37) indicate "(the Entity Relationship model) …is a high-level conceptual data model …we define a data model as a group of concepts that help us specify the structure of a database and a set of associated operations for specifying retrievals and updates on the database".

With this support, a Tactical Model of Development was stated, based on the selection of an Entity (Step 3 above) to focus upon (thus "Focal Entity") and to elaborate through all of the various appropriate models – Conceptual (Entity Definition), Logical Data (Table Definition), Physical Data (Table Construction), Process (Forms and Reports).

Concepts of Entity "cohesion" enabling the focus upon a single entity, identification of an entity,

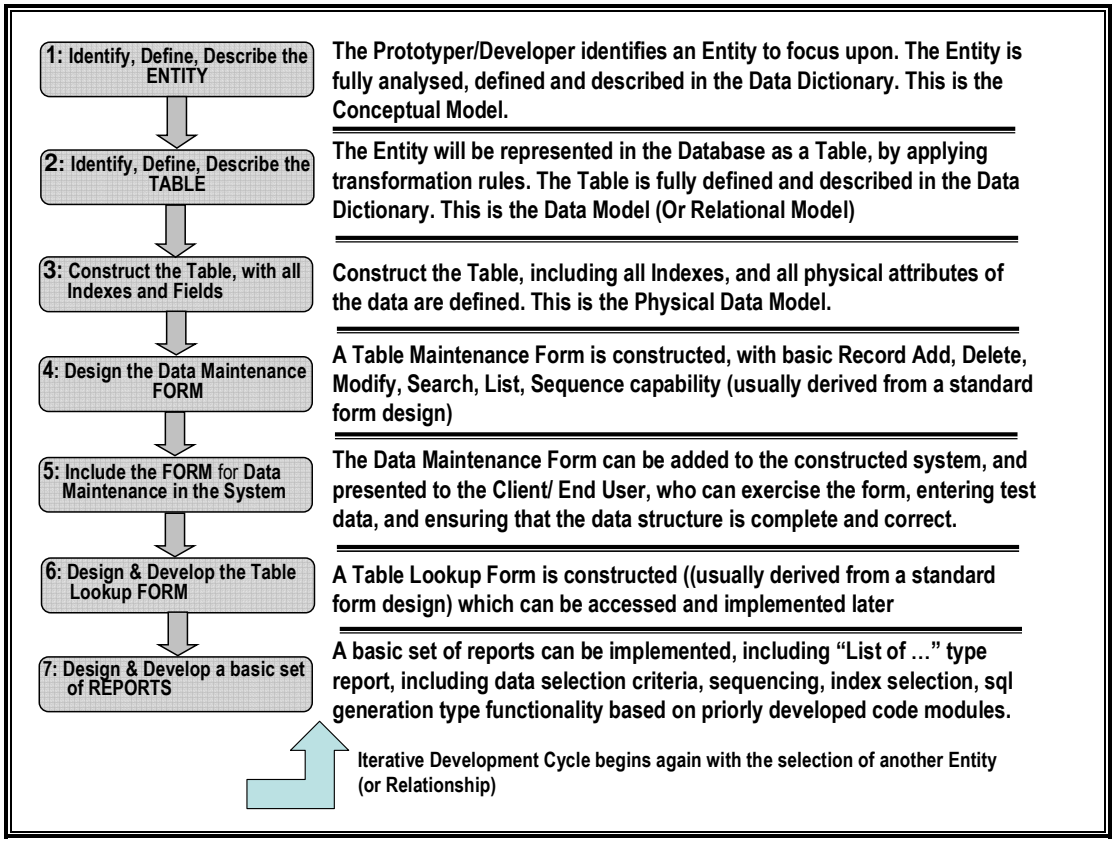

Figure 2: Tactical Model of Focal Entity Prototyping (Source: Morien, 2003a)

simplification of the ER Modelling rules etc. are elaborated in a variety of unpublished discussion papers (see Morien 2003a, 2003b, 2003c, 2003d)

This approach to development drew together a number of relevant and important streams of research, discussion and practice to create what is seen as a coherent approach to development in the "rapid development / agile" mode (although the term "agile" has more recently been added to this lexicon). First, there was the "Prototyping" stream, wherein a highly iterative, visual approach to development was mooted. Then, partly within the "prototyping" approach, but independently as well, reliance on the availability and use of a large variety of software development tools and technologies, including Data Dictionaries and Code Generators. In addition, drawing on previous work by Jenkins (Jenkins, 1984) and Naumann & Jenkins (1983) where prototyping was seen to be "data-oriented", the author's own expertise and interest in Entity Relationship Modeling was introduced into the equation. Elements of Information Engineering (Martin & Finkelstein, 1981), especially CRUD Analysis (Create, Retrieve, Update, Delete process analysis, as devised in the Information Engineering approach to systems development (Martin & Finkelstein, 1981), implying process modeling as part of the ER Modeling activity, were co-opted. The final piece of the puzzle is a heavy reliance on standards and reuse; standard form designs that are reused, standard documentation templates and styles that are reused, standard processing requirements etc.

The outcome is an iterative, prototyping approach, based on the identification of entities (and relationships), using high productivity development tools to incrementally develop a database processing system, including the database itself. Fully functioning parts of the system are able to be delivered to the Client at frequent and regular intervals for verification and validation.

As with any system development, some up-front analysis must be undertaken. How much analysis, and how complete that analysis must be, is the major point of division between the supporters of the Heavyweight/Waterfall Model development methodologies, and the alternative Lightweight/Evolutionary development methodologies. This model commences with the creation of a Working Entity Model, the creation of which is partly justified by Pareto's Principle that indicates, in this situation, that 80% of the identifiable entities are able to be discovered very quickly (in 20% of the effort?). There is no requirement or principle at this point that each or any entity is a "must", or that any or all entities must be totally and fully defined, especially as to its attributes.

Managing Projects with SCRUM & Focal Entity Prototyping

This "Focal Entity Prototyping Approach" fits extremely well with the concept of the SCRUM "sprint". The "Tactical Model" may well be the basis for allocating the sprint tasks to the Personal Sprint Backlog, and the outcomes of this model are the contracted outcomes of the sprint. The selection of a specific entity or relationship, and carrying through the development cycle stated in the Tactical Model, is a well-defined, discrete "chunk" of the system that is very well stated as the target deliverable of a sprint.

It is interesting to see the equivalence between this approach, in a sprint, with the Extreme Programming concept of a "story", and other approaches to defining a deliverable increment to the system, in an iterative cycle. In Extreme Programming, a User Story represents a feature of the system. The customer writes the story on a note card. Stories are small. The estimate to complete a story is limited to no greater than what one person could complete within a single iteration. (Object Mentor, 2003).

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
- 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),
- 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 time-sheet, that referenced these outcomes. So a Plan – Act – Reflect cycle was established.

End of Project Questionnaire

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

By these various means, a substantial amount of information about the students' activities, personal views, reflections, opinions, complaints and achievements was elicited. As the information was collected in a variety of ways, over an extended period of time, it is felt that the feedback was not contrived for "assessment purposes only", and did accurately and appropriately reflect the students' views over time, including reflecting the changing views as the students became more experienced in the project activity and all its methodological and technological aspects.

Evaluation and Discussion

Is Incremental Development Really Proper?

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). There was a definite "This can't be right?" attitude amongst many students. This was manifested by many groups who would not adhere to the principles stated in the Tactical Model of Development. This model clearly implies a single thread of development from the conceptualization of an entity through to the construction of a table, and the creation of a set of processing elements specifically for that entity. This activity can be undertaken by a single developer, in parallel with other developers, who may be doing exactly the same thing on another entity. The outcome of this is, inter alia, a well-defined and well described single entity, with all appropriate documentation, and applicable processing.

However, many groups insisted on attempting to push far ahead of construction by developing a "full" Entity Relationship Model. The idea that the whole ER Model did not need to be completely defined as a precursor to any other development activity was very difficult for the students to accept. Unfortunately, there are discussions in the literature wherein prototyping is seen as an activity that commences after a full database design has been completed. This is essentially an interface prototyping approach. Also, prototyping is sometimes seen as inevitably "throwaway". The variety of views on and approaches to prototyping are well discussed in McConnell (1996), where some of these approaches are elaborated as rapid development "best practices".

There was certainly an attitude amongst students that "more is better", and many students indicated that they would "go back and fix it all up" at the end of the project, thus creating what the author termed a "debris field" of partially completed and often buggy deliverables. The term "debris field" was used in various TV documentaries about discovering shipwrecks, and the author felt it particularly appropriate in the circumstances. The idea of completing a small part of the system to absolute, fully tested and validated completion was hard for some students to accept, especially if they met difficulties in doing that. To demonstrate progress, they would abandon this piece of the system, and move on, intending to return later to "fix it all up."

There was also some resentment by students who perceived that they had been "forced into using these methods". Some students felt, early in the project, that their "creativity" had been limited by having an agile / prototyping approach "imposed" upon them. One student quote is "Fewer influences by the lecturer on development would be optimal to support a creative involvement."

There were some rather famous group cases where the group fought every inch of the way through the first semester to avoid doing things this way, but who finally realized how badly organized their efforts were, and adopted a very orderly and well-stepped approach, using the Focal Entity Prototyping Approach, to effectively rescue their project from disaster in the second half of the project.

One group (OST) stated "…we didn't really use a development methodology; perhaps the waterfall model was followed roughly. This approach left us behind in development and without a sense of direction…. (subsequently) we decided to look more closely at the Focal Entity Approach". It can be reported that this group pulled itself out of its difficulties and produced a very good outcome.

Another group, that ultimately failed, defined a "Rapid Application Development" approach that consisted of "Planning ➔ Design ➔ Development ➔ Cutover". Their further comments in-

cluded "We did not really plan to use focal entity prototyping at first … sometime businesses have a lot of entities, we might end up with a lot of entities and this can be quite complicated to do." This group patently failed to see that an orderly, incremental approach, based on the entity as the "chunk" or increment to be developed, clearly acknowledges the problem of "a lot of entities", and would overcome this problem by a simple, stepwise and iterative development, based on each entity in turn.

Certain questions were asked in the End Of Project Questionnaire in regard to this point.

Development Approaches Adopted

As shown in Table 1, some groups indicated a combination of approaches. This classification is taken as much as possible from the students' own words. However, it can realistically be said that all of these approaches come under the general heading of "agile" development methods. The more specific term is Focal Entity Prototyping Approach". The Unit Leader was keen to establish this terminology and practice as relevant and useful.

Table 1: Analysis of Question: Development Approach used by your Group?

▪ **Prototyping**	**9**
▪ **SCRUM / XP (specific Agile approaches)**	**12**
▪ **Focal Entity Prototyping Approach**	**12**
▪ **Rapid Application Development**	**4**

Choice of Development Approaches Adopted

As shown in Table 2, there were multiple responses to questions about choice of development approach. For example, some groups indicated both "The Unit Leader required that we do it that way" and "We thought that it was best….".

Table 2: Analysis of Question: Why did you choose that Development Approach?

Methodology Indicated ➔	Prototyping	SCRUM, XP	Focal Entity	RAD	TOTAL
▪ **The Unit Leader required that we do it that way**	2	2	3		7
▪ **The Client required that we do it that way**		1		1	2
▪ **We had previously been taught that way**	3	2	4	3	12
▪ **We thought that it was best for the type of system we were developing**	4	7	5	1	17

The fact that only 30% of students indicated that "We had previously been taught that way" supports the statement that most students did not really have prior knowledge of these approaches. This is especially seen when we consider the responses in Tables 3 and 4.

Some groups did not answer all of the questions. However, there is a clear indication here that well over 50% of students had no knowledge whatever of this development approach.

Table 3: Analysis of Question: Prior to this project, my knowledge of "Agile Methods" was:

▪ **Excellent**	**1**
▪ **Good**	**1**
▪ **OK**	**10**
▪ **Non-Existent**	**16**
▪ **I don't know what "Agile Methods" are.**	**1**

Table 4: Analysis of Question: Having done the project, my knowledge of "Agile Methods" is now:

▪ **Excellent**	**1**
▪ **Good**	**13**
▪ **OK**	**8**
▪ **Non-Existent**	
▪ **Confused – I cannot see the overall approach clearly**	**1**
▪ **I still don't know what "Agile Methods" are.**	

It is considered to be one of the highly successful learning outcomes of the projects that the students generally learned about these methods. It was interesting to note that the sole group that responded "Confused - I cannot see the overall approach clearly" indicated that they did not in fact adopt an agile approach.

Further analysis of the student responses and opinions indicates that, after some initial refusal, and confusion, students generally found this approach to be most acceptable.

It is the intention of the Unit Leader, for the next cohort of project students, to prescribe the book "Lean Software Development: An Agile Toolkit" (Poppendieck & Poppendieck, 2003) to formalize the adoption of an agile approach.

The students were asked to indicate their views on the usefulness and outcomes of using these agile approaches. This was done in a guided fashion (i.e. pick list) and in open comments.

Question: Adoption of a "Waterfall Model": Discuss ……

Almost every group indicated that they had not ultimately adopted a "waterfall model" of systems development, for a variety of reasons. Obviously the influence of the Unit leader in mandating, where appropriate, that an agile / lightweight approach be adopted, was a significant influencing factor. But what did the students have to say about the Waterfall Model and SDLC method?

> *"We feel that this approach is not flexible"*
>
> *"Too much emphasis on documentation 'up front'"*
>
> *"A high level of client interaction is needed, that is not allowed by the waterfall approach"*
>
> *"The prototyping approach proved so useful that to adopt a waterfall approach would have been quite useless"*
>
> *"Didn't have enough time to adopt a waterfall approach"*
>
> *We, as a group, did not adopt the Waterfall Model Approach … we did not possess the business analysis acumen to enable us to fully comprehend a set of system requirements, which are in a state of flux generally, and set about constructing a successful system…. Another downfall of the Waterfall Model Approach is the lack of feedback from the client. We knew that we would value the client's feedback and this was indeed the case. A large amount of expertise would be required to model the system from scratch."* (EyeQ Group)

Two major observations by students arise from these comments. First, the waterfall approach is seen to be much more time consuming, and indeed it is. Research and personal experience of the Unit Leader indicates that by the application of a well-structured prototyping approach, develop-

ment times and budgets can be halved. (Morien & Cant, 1994). The second observation is that the students realized the benefit of an incremental, evolutionary approach as a learning mechanism whereby continuous education for both user and developer took place during the entire development period. This is not achievable in the Waterfall Model, where, after the Analysis phase, the user is not longer involved until the system is implemented; often to the distress and disappointment of the user, and the rejection of the delivered product. This failure to produce a useful, useable and acceptable system has been demonstrated in an overwhelmingly frequent number of cases, with associated costs and waste calculated in the billions of dollars. (Standish Group, 1994)

General 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"*
>
> *"helpful … forces us to look forward, focuses on how to better the system"*
>
> *"it enabled us to more quickly determine requirements"*
>
> *"early learning was helpful in later iterations"*
>
> *"ensures that the client's requirements are met"*
>
> *"quicker"*
>
> *"…we were also able to present to our client each entity and for our client to provide us with any feedback or on any required changes. This has also allowed our client to provide us with information about this single entity instead of dispersing his focus to the whole environment. We were then able to efficiently iterate through each cycle, producing a system that is built upon our client's requirements."* (ZAP Group)
>
> *"At first we found this approach to be time consuming and frustrating but once we adopted it, we found it very effective."* (OST Group)

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

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 over 28 years' experience, 19 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.

As shown in Table 5, 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.

Table 5: 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**	**32**
▪ **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**

An excellent outcome.

Adoption of Standards, and a Reuse Policy

Students were asked to discuss the policies and practices that the group stated, and practiced, in regard to 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 … although it was difficult to create standards, it ultimately saved us considerable time".*
>
> *"It is extremely efficient…"*
>
> *"It is quick to adopt, and really saved us time"*
>
> *"It helped us shorten the development time and reduce debugging and testing time"*
>
> *"Reuse was facilitated by the Focal Entity Prototyping Approach … it took 15-20 minutes to create another form from the standard"*
>
> *"Based on our research, information gathered from our Unit Controller and our own experience with Reuse, yes, we have discovered that it is an effective approach as we were able to reduce the amount of time in coding and analyzing how to go about solving a similar problem. In addition, reusing these components produced less errors because members did not have to build complex chunks of code again as it may already be available for use."* (ZAP Group)
>
> *"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 have 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)

Some specific outcomes of the adoption of a policy of standards, standardization and reuse, were elicited (see Table 6).

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.

However, most students came to very satisfactory realizations about this, and the figures in Table 5 clearly indicate that standards and reuse variously enhance group harmony, productivity and quality to a significant extent.

Table 6: Analysis of Question: Having well stated standards was beneficial to the good order and efficiency of your system development activity?

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	1	2	6	12
• Productivity			3	9	10
• Quality Outcomes			1	13	10
• Acceptability by the Client			3	9	10
• Enhancing member's knowledge		1	5	9	7
• Other (Time taken to develop some functions)					1

Differentiation of Models

As shown in Tables 7 and 8, many students failed significantly to see the difference between the Conceptual Model as manifested as an ER Model, and the Logical Data Model (or Relational Model, depending on your preferred terminology).

Table 7: Analysis of Question: I now fully understand the modeling activity based on a clear difference between the ER Model and the Logical Data Model.

▪ Yes I do, very clearly	12
▪ I do, but it is still not clear	8
▪ I just cannot see the realistic difference	
▪ I can't really understand this question enough to answer it.	4

This matter of differentiating between models is considered to be very important, especially as we move towards "model driven architectures" and "model driven development" (OMG, 2003). The Unit Leader is attempting to ensure that the students understand the clear differentiation between the Conceptual Model, manifested in this case as an ER Model, the Logical Data Model, which is extracted from the ER Model using a simple set of transformation rules, the Physical data Model, which is designed to most efficiently use a specific DBMS's data design and storage mechanisms, and the Processing Model.

An interesting aside is worthwhile here. In another unit, this author presents to the students an industrial-strength database that is actually in use in a sales organization. The database encompasses inventory management, customer management, sales, invoicing, payments etc. There are over 65 tables. The students are required to backward engineer this database into a data dictionary. This is often a significant chore for the students, who fre-

Table 8: Analysis of Question: In regard to the "ER Model is different to the Logical Data Model" viewpoint proposed by the Unit Leader, my viewpoint on this is:

▪ Viewing them as different is an excellent and simplifying approach	14
▪ I see absolutely no point in seeing them as different	4
▪ Having the 2 models just doubles my work, and duplicates things.	4
▪ I can't really understand this question enough to answer it.	

quently complain about the lack of information as to the content and structure of the provided database, and often ask the question "what does it mean that ….?" In other words, the data is there, the structure is evident, but the semantics or meaning of the data, the essential business rules and business roles and scenarios are missing. This is the stuff of the ER Model. The students are coming into this situation at the data structure level, not the semantic and business requirements level. The absence of the Entity Relationship Model, stating the semantics and rules about the information architecture discovered, and the role of the Entities and Relationships in the business processing, and presentation of business processing scenarios, left the students bewildered.

Too frequently it seems Entity Modeling is confused with Data Modeling, or they are seen as one and the same thing. Textbooks frequently refer to Data Diagrams showing entities and relationships. It is interesting to interrogate these students and ask them 2 simple questions; "What is an Entity?" and "What is a Table?" The obvious difference is stated, which leads to the inevitable question "Then how can you put them both on the same diagram and say that they are the same thing?"

It is believed that one motive for not liking this differentiation is that the students see it as imposing an extra documentation load on them, although this was not reflected in the responses shown in Table 8 above. Why define entities, and then define exactly "the same" tables? Why define attributes, and then duplicate that list by defining data fields?

There is more than a modicum of sense in this view, and the principle of "one write, use many" should be applied, wherever possible. The use of appropriate development tools and technologies would be of great assistance here, and it is proposed to develop a simple data dictionary system that will allow automatic and semi-automatic transformations from one model to the next, and to produce the documentation as a reporting outcome, rather as the activity. It is obviously more productive to have a live data dictionary, rather than ultimately merely create a static document.

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 have 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 substantially 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?

Two groups said Yes. However, both of the client organizations for these groups were technology companies, and the personnel involved were experienced system developers. Every other group said an emphatic No.

Student comments included:

> *"The client cannot envisage the final product"*
>
> *"It would be nice to receive all this information, but doesn't seem possible. In the first phase there is a lot of information to get through, and things will always get over looked. Also as the system is being developed the client may realize that they have forgotten to mention something, or they would like some extra features included."* (Chevron Group)

> *"As (the client) got to see the progress of the prototype and we proposed added functionalities, and she was happy with those, she came up with new ideas/functionalities as to what she wanted the system to do."* (OZMAU Group)

> *"We don't believe that this was possible as the client didn't appear to understand the current system or all the requirements correctly, thus it was hard for them to convey them to us. The complexity of the system also added to this. It took many client meetings to work this out."* (OST Group)

> *"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 have continuing opportunities to learn about requirements, about possibilities and options, at a time when those realizations can be incorporated in the system. Also, as the project proceeds, both client and developer get better 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. Their comments are informative in this regard.

> *If the Client had proceeded to overwhelm us with all of the details at too early a stage that would have been most confusing and contradictory to the efforts of understanding the system we were to build. As a group we did not know enough about the business processes during the initial stage of the development to be given all of the details. The Client was very focused on the project and willing to offer as much information about the requirements for the system as possible."* (EyeQ Group)

The author had the opportunity to sit in on group presentations 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. But 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:

"Project progress and features were visible to the client, and this ensured that quality remained high"

"Yes, with demonstration of prototypes on regular basis, client can see the improvement and have 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 have 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 alright 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 have 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 9 reflects the opinion of students about presenting the prototype to clients early in the project.

Table 9: 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**	**18**
▪ **NO, the Client wasn't impressed.**	**1**
▪ **Can't Say – the Client has not indicated anything about this**	**4**

Conclusion

Nearly every one of 135 students gained significant knowledge and experience of agile development methods. 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 30 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.

They are, however, empowered in this by the highly visible nature of the prototyping approach which acknowledges this situation, and 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, as many managers may fear, is not invalidated or put at risk. 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.

In regard to the Focal Entity Prototyping Approach, which was a central theme in many of the projects, it has been demonstrated that it provides a highly successful project framework. Success in this regard is defined as having and using a well-defined development approach that enhanced the project development activity by providing guidance to the project members by a well-ordered, well-structured and simple step-wise approach that allowed the project group to proceed in a highly productive, orderly fashion to produce appropriate deliverables,

All groups who were developing an on-line, interactive database processing system, and who adopted and adhered to this approach, were very successful in their project outcomes. Those groups who did not were often not successful.

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 that "The adoption of Agile Development Methods for system development is appropriate and leads to project success". With feedback from over 30 groups, encompassing the views, opinions and experience of over 130 students, it can be at least tentatively stated that this hypothesis has been proven.

I believe that the evidence is clear. Nearly unencumbered by the baggage of experience, these students have given a resounding Yes to these new, contemporary approaches, and a distinct NO to the traditional rigorous pre-definitional development approaches, as manifested in the "Waterfall / SDLC" approaches that 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.

Our incoming professionals, the graduating students, have seen it, and have done it. We certainly hope that they can and will influence system thinking in the years to come.

References

Agile Alliance. (2003). Retrieved December 12th, 2003, from http://www.controlchaos.com/

Appleton, D.S. (1983, November). Data-driven prototyping. *Datamation*.

Bemelmans, Th.M.A. (Ed). (1983). *Beyond productivity: Information systems development for organisational effectiveness.* North-Holland.

Berghel, H. (1994). New wave prototyping: The use and abuse of vacuous prototypes. *ACM Interactions*, *1* (2), 49-54.

Budde R., Kuhlenkamp K. et al. (1984). *Approaches to prototyping.* Springer Verlag.

Burns R.N. & Dennis A.R. (1985). Selecting the appropriate application development methodology. *Data Base*, Fall.

Cockburn, A. (2003) *Crystal methodologies*, Crystal Main Foyer. Retrieved December 14th, 2003, from http://alistair.cockburn.us/crystal/crystal.html

DSDM Consortium. (2003). Home Page. Retrieved December 14th, 2003, from http://www.dsdm.org/

Elmasri R. & Navathe S.B. (2000). *Fundamentals of database systems* (3rd ed.). International Thompson Computer Press.

Extremeprogramming.org (2003). Home Page. Retrieved December 14th, 2003, from http://www.extremeprogramming.org/

Fowler, M. (2000, December). Put your process on a diet. *Software Development Magazine*.

Fowler, M. (2003). *The new methodology.* Retrieved December 14th, 2003 from http://www.martinfowler.com/articles/newMethodology.html,

Gilhooley, I.A. (1986). A methodology for productive systems development. *Journal of Information Systems Management, 3* (1).

Hawgood, J. (Ed.) (1981). *Evolutionary information systems.* North-Holland.

IBM. (1999). *IBM Research Journal.* Retrieved December 13th, 2003, from http://www.research.ibm.com/journal/sj/381/millet.html

Jenkins, A. M. (1983). *Prototyping: A methodology for the design and development of application systems.* Indiana University.

Jenkins, A. M. & Fellers J. (1986). *An annotated bibliography on prototyping.* Indiana University, IRMIS Working Paper W613.

Martin, J. & Finkelstein C. (1981, November). *Information engineering.* Savant Institute Report.

McConnell, S. (1996). *Rapid development: Taming wild software schedules.* Microsoft Press.

Morien, R (1993). Prototyping large on-line systems: A focal entity approach. *3rd Australasian Conference in Information Systems.* Wollongong University, October.

Morien, R. (2003a). *Simplifying the entity modeling activity – A simple set of rules to follow.* Unpublished paper, School of Information Systems, Curtin University.

Morien, R. (2003b). *Identifying entities, relationships and attributes*. Unpublished paper, School of Information Systems, Curtin University.

Morien, R. (2003c). *Representing entities, relationships and attributes in the relational model.* Unpublished paper, School of Information Systems, Curtin University.

Morien, R. (2003d). *Visualisation of the entity model: Visual development methods for understanding the entity modeling process.* Unpublished paper, School of Information Systems, Curtin University,

Morien, R. & Cant, R. (1994). Specification with prototypes: Two case studies. *Proceedings of the 5th Australian Conference on Information Systems*, Monash University, September.

Naumann, J.D. & Jenkins, A.M. (1982). Prototyping: The new paradigm for systems development. *MIS Quarterly, 6* (3).

Object Mentor Inc. (2003). Home Page. Retrieved December 14, 2003, from http://www.objectmentor.com/processImprovement/index

OMG (Object Management Group). (2003). Retrieved December 14, 2003, from http://www.omg.org/mda/

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

Poppendieck, M. & Poppendieck, T. (2003). *Lean software development: An agile toolkit*, Addison-Wesley, Pearson Education.

Simsion, G. C. (1994). *Data modeling essentials: Analysis, design and innovation.* van Nostrand Reinhold.

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

Standish Group. (1994). *The Chaos Report (1994).* Retrieved December 14, 2003, from http://www.standishgroup.com/sample_research/chaos_1994_1.php

Svetinovic, D. & Godfrey, M. (2003). A lightweight architecture recovery process. Software Architecture Group, University of Waterloo. Retrieved December12th, 2003, from http://plg.uwaterloo.ca/~migod/papers/swarm01.pdf

Sumner, M. R. (1985). How should applications be developed: An analysis of traditional, user, and microcomputer development approaches. *Data Base*, Fall.

Vacca, J. R., (1984). Evolutionary systems development: An overview. *Systems Development Management.* Auerbach, Dec/Jan.

Biography

Roy Morien has been a lecturer in the School of Information Systems, Curtin Business School, Curtin University of Technology, since 1985. Prior to this he held a number of IT positions in the Australian Public Service, some at senior project manager level, and in private industry, having commenced his IT career in early 1977, as a trainee analyst/programmer with NCR. Roy's primary research interests are in the productive development of computerized business information systems, encompassing development technologies, rapid and agile development methodologies, prototyping, and more recently Model Driven Architecture.