2001

Improving Software Development: The Prescriptive Simplified Method

Ilona Box
University of Western Sydney, i.box@uws.edu.au

Jeff Ferguson
University of Western Sydney, j.ferguson@uws.edu.au

Improving Software Development: The Prescriptive Simplified Method

Ilona Box[a] and Jeff Ferguson[b]

[a]School of Management
University of Western Sydney, Quakers Hill, Australia
i.box@uws.edu.au

[b]School of Computing and Information Technology
University of Western Sydney, Parramatta, Australia
j.ferguson@uws.edu.au

Abstract

We describe the Prescriptive Simplified Method (PSM), a software development method, based upon UML and use cases. PSM uses a traditional software development cycle and embeds use case development in the system construction phase. PSM has been used in three pilot studies. These studies support PSM as a viable means of transitioning lay people to novice system developers. This paper defines PSM and a teaching approach. Our overall goal is to validate both by tracking the development of a cohort of students, from the beginning of their training through to their early experiences in industry.

Keywords

IS education, IS training and development, IS development methods and tools, IS development strategies, IS life cycle activities, IS skill requirements

INTRODUCTION

There are several motivations to our work. There is a shortage of IT professionals; this has been well reported in scholarly and popular press. Mass education has meant a change in the type of student at university (Biggs 1999). There is a need for graduates to have good software development skills (Hellens et al. 2000). Our experience with IS development methods available in texts has been that they are targeted above the ability of the majority of the students. These factors lead us to develop a new method the Prescriptive, Simplified Method (PSM) and a teaching approach for it.

Jayaranta reported that, in 1994, there were over 1,000 brand name information system development methodologies worldwide. The advent of the Unified Modelling Language (UML) as a standard has made the debate for object-oriented over structured methods more persuasive. The combination of UML with use cases has great practical value (Constantine and Lockwood 1999, Jacobson et al. 1992, Larman 1998; henceforth referred to as earlier use case approaches). These methods are thorough and quite complex.

So why develop yet another method? When teaching these earlier use case approaches to lay people we had a number of difficulties. The earlier use case approaches, and the associated texts, often assume prior knowledge of information systems development. Students found some approaches ambiguous about when use case development starts, when the project starts, what is delivered at the end of use case development and at the end of a project. It was difficult to cover the knowledge of the business view and the system developer view while also teaching the complexities of these approaches. When teaching earlier use case approaches, we taught that each use case should deliver an increment useable by the client. However, in practice, the use cases need to be integrated into an application framework to create a product. Also, good programming practices such as pseudocode, modularisation and program structures still apply in object-oriented development.

The above difficulties led us to develop the Prescriptive Simplified Method (PSM) to transition lay people to novice developers. PSM is specifically targeted at developing software applications for business (as opposed to, for example, control systems), which allows for considerable simplification over earlier use case approaches. In earlier use case approaches, use case development is diffused throughout the entire method. In PSM, use case development is explicit and localised. PSM uses a traditional software development cycle and embeds use case development in the system construction phase. We do not claim that our method is of industrial strength. (For example, we have not included packages or components to deal with networking systems.) Rather it is the place to start for a lay person so that he/she can become a novice system developer. Once novices gain more experience we expect that they will be able to adapt to any new development method they may encounter.

This paper will describe our research method, the pilot studies that contributed to the refinement of PSM, the PSM and the teaching approach we have used.

RESEARCH METHOD

This paper describes the first of four stages anticipated to take several years.

Stage One

Stage one is largely complete and took two years. Stage one was exploratory and developmental; the major outcome being the PSM, which was initially developed and then modified during pilot studies. We initially developed PSM from our experience with other methods, structured and object-oriented. The first two pilot studies have resulted in some refinement of the method. We are refining the PSM and developing the teaching approach in a third pilot study.

Stage Two

Stage two will involve the instruction of lay people in the PSM, and measuring the student outcomes. (In fact, PSM is currently being taught, but as the teaching materials and techniques are under development, the transition of the students from lay people to novice developers is not being measured.) The instruction is bottom-up, in two components.

1. In the first component, students learn the tasks, tools and techniques of PSM.
2. In the second component, students learn the structure and means of producing project deliverables.

To assess and measure the change in the knowledge and skill level of the individuals participating in stage two, prescribed tasks on two set projects and questionnaires will be used. Interviews will be conducted with a subset of the stage two group to gather qualitative data about the instruction mode and PSM.

Stage Three

During stage three the novices from stage two will be grouped into teams of two or three people. Each group will be assigned a (non-critical) project from a genuine enterprise outside the university (henceforth referred to as "the client"). An experienced developer will supervise the groups during the 24 to 33 weeks available for each project. Supervision will be directed toward assisting project management and organisational behaviour issues, not remedial instruction in PSM, nor direct involvement in the actual development of the product. The supervisor will be interviewed to discover if additional instruction about the PSM took place. Clients will be interviewed during the project. At the first interview, their prior experience with information systems development will be established. At subsequent interviews, clients will be asked to identify where the project is placed in regard to PSM. Student groups will be interviewed to discover any difficulties experienced with PSM such as gaps in knowledge or skills and ability. The quality of the deliverables from the groups and of their respective final product will be assessed. Sufficient data is to be gathered that will correlate the quality of the product, the PSM, client satisfaction and skill level of the novice system developers.

Stage Four

Stage four will focus on individuals from the groups in stage three who have entered into system developer roles in industry. The individuals and the employers will be interviewed a number of times over a year. Evidence of the further use of PSM by the former students will be gathered. An assessment of the employer's and individual's skills in object-oriented information system development will be made during the year, with two goals. First, to compare change in skill level, of both the former student and their organisation. Secondly, to assess the influence of the former student on any organisational change, and to link that change to PSM.

PILOT STUDIES

The aim of the pilot studies was to test the completeness of the description of PSM. Completeness in this context meaning that the materials about PSM provide enough information for the students to understand, comprehend and apply the method.

In the first pilot, students applied the PSM to develop, in collaboration, a single project. The project was developed up to the point of full specification prior to coding. (The pilot was similar to the research method stage two, second component as described in the previous section.) A group of 40 students were taught over 16 weeks. The prior skill levels of the group members were diverse. Some had no prior background in object-oriented tasks and techniques. Just over half the group had received instruction in structured methods. The remainder of the group had received instruction in an object-oriented development method based on the text by Larman (1998). (Larman assumes his audience has prior knowledge of information system development.) The entire group was instructed on object-oriented software development in the context of PSM. This pilot resulted in some changes to the PSM. The changes included additional clarity of the specification of entry and exit criteria at a level comprehendible by the students. This first pilot will be repeated with a group that will have more homogeneity of knowledge and skill with PSM.

The second pilot of PSM occurred with two people over 8 weeks. This pilot focused on the stage two, first component, that is, the instruction in the tasks and techniques of PSM. Instruction typically occurred for twice a week for two hours at each session. Each individual also sought additional instruction to clarify the business requirements of the project and to monitor the accuracy of his/her completed tasks. This pilot resulted in minor PSM refinements to the works describing the tasks and techniques. (Space does not permit the inclusion of all these instructions). The refinements mainly added clarity and finer detail to the descriptions of the techniques.

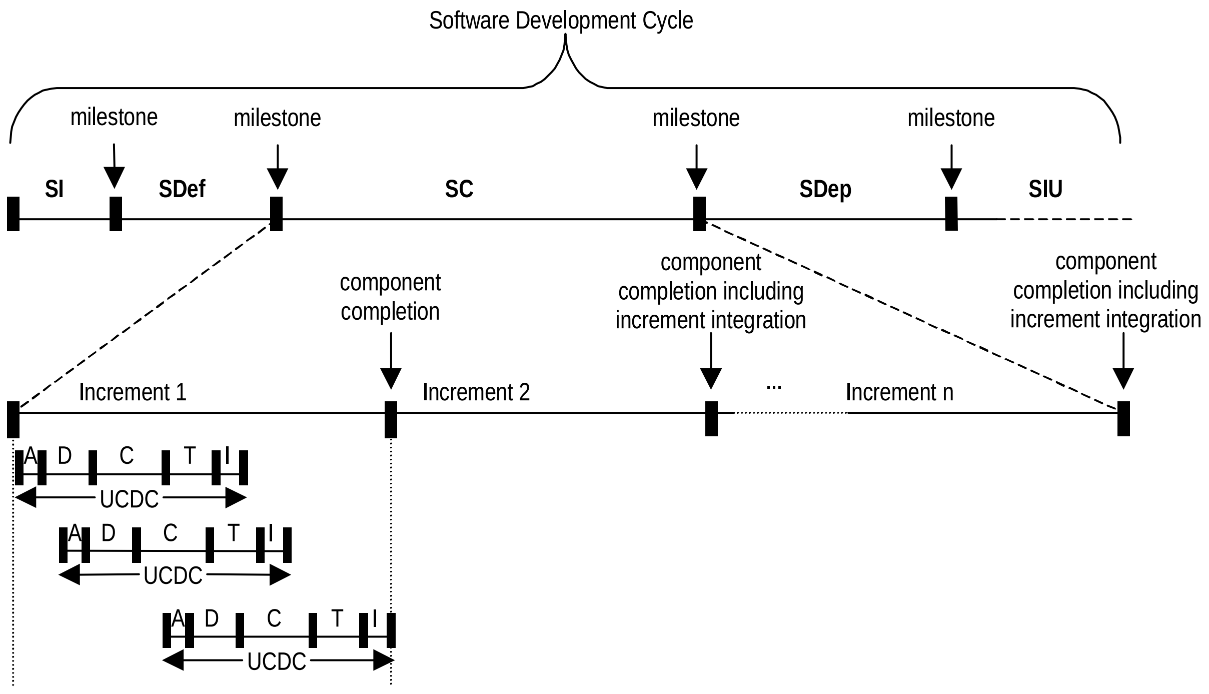
At the time of writing, the third pilot of PSM is in its twelfth week of a thirteen week study. This pilot commenced with 93 participants. It is anticipated that about 50 participants will complete the study. This pilot repeated and overlapped the second pilot. Instruction typically occurred once a week for three hours. A small number of the total group sought additional instruction to clarify the business requirements of the project and to monitor the accuracy of his/her completed tasks. So far this pilot has resulted in the discovery that the case study describing the exemplar project needs to detail all business requirements so that they can be mapped to tasks and models in the PSM.

One significant outcome of the pilot studies was the need for complete and relevant examples of all of the tasks, entry and exit criteria. We believe that lay people need the examples as they have very little vision about what their work will become and have little ability to imagine the final product. A completely worked project will accompany PSM with copious annotations between PSM instructions, the exemplar project and the developer's working papers.

THE PSM

PSM (Figure 1) is a software development cycle (SDC) with embedded use case development cycles (UCDC). PSM phases are system investigation (SI), system definition (SDef), system construction (SC), system deployment (SDep) and system in use (SIU). Each UCDC has five phases: analysis (A), design (D), construction (C), testing (T) and integration (I). Tables 1 and 2 indicate the entry and exit criteria of the SDC and UCDC phases, respectively.

Figure 1 The Prescriptive Simplified Method


Software Development Cycle
milestone
milestone
milestone
milestone
SI
SDef
SC
SDep
SIU
component completion
component completion including increment integration
component completion including increment integration
Increment 1
Increment 2
...
Increment n
A
D
C
T
I
UCDC

The specification of entry and exit criteria does not preclude overlap from phase to phase nor the discovery of knowledge about the project relevant to other phases or tasks in the project. The PSM encourages the use of the note notation, specified in the UML, to record discoveries.

During the SDef phase, high level use cases are identified, categorised and ranked. Each use case identified in the SDef phase is dealt with in rank order. The development of the highest ranked use case may commence before the SDef phase is completed. In the SC phase, a use case may be developed as one increment, or be integrated with other use cases to form an increment. Each use case has its own UCDC. UCDCs may run consecutively, or have concurrent or staggered starts. The first pilot had thirteen use cases, the UCDCs had concurrent starts. The three use cases developed during the second and third pilots had staggered starts. Figure 1 shows three UCDCs with staggered starts. As each UCDC is completed the executable component is integrated with the executable component from previous use cases.

In earlier use case driven approaches, use case development is diffused throughout the entire method. In PSM, use case development is explicit and localised. PSM uses a traditional SDC and embeds the UCDC in the SC phase. Use case development in PSM is instigated after the high-level use cases are identified and ranked. Because PSM uses an SDC, the client/manager (who may have no background in systems development) has a clear and linear view of the project. Progress can be measured and delineated by the milestones at the end of each phase. Only the development team need be concerned with the iterative development of use cases and the overlap of tasks from one phase to the next. The user(s) of the function(s) delivered by any one use case need only be involved in the UCDC for that use case.

The description of all phases of the SDC and UCDC has been standardised. The repetitive nature of the layout, we believe, helps students anticipate what to look for and where to look while learning PSM. Figure 2 shows the template for describing any PSM phase. Section 1.1 ("Purpose of the Phase") presents the context of the phase. It defines when the phase starts, whether it is part of a UCDC or the SDC. The ambiguity about starts and ends of use case development and project development is removed.

Section 1.2 ("Responsibilities") defines the roles of project participants. Our experience prior to PSM was that students had difficulty separating the roles and apportioning tasks among project participants. For example, the students often neglect the user as the source of requirements after the initial phase of the development cycle. With judicious choice of learning tasks, students use this section to overcome these difficulties. This section reinforces the need for the project to have management approval as each milestone or phase exit criterion is met. Both the developer and business views are discussed and contrasted in this section.

Section 1.3 ("Software Development Method Tasks") contains the tasks for each phase. Our choice of tasks is a limited set that can be used to develop a phase from start to finish (as described in tables 1 and 2). The set is not all the available models in UML. We intentionally limited the set to one that covers system development in a simplified way for the development of business information systems (and not, for instance, control systems). Our method is intended for educational and industry training environments. It is the place to start for a lay person, such as a student or new recruit, so that he/she can become a novice system developer. A novice system developer would have sufficient skills to be a worthwhile contributing member of a software development team. Once novices gain more experience we expect that they will be able to adapt to any new development method they may encounter. Also, because of the standardised layout of PSM, the novices will know what to look for in any other methods they encounter.

Phase	**Entry Criteria**	**Exit Criteria**
System Investigation	Some form of proposal. It may be anything from a verbal request to a written document that has the vote from the board of directors of the organisation.	Statements of • The Problem • The Client • The Business Functions • The Risks • The Business Case • The Project Plan, in general with more detail about the System Definition Phase Plan
System Definition	System Investigation Phase output, plus management approval and resource commitment.	• High level use cases • Use case diagram • Ranked use cases (categorised as core or adjunct) • Analysis Class Diagram • Project development plan (based on use cases) • Detailed plan for System Construction Phase (based on use cases) • Management/owner/user agreement
System Construction	System Definition Phase output, plus management approval and resource commitment	For the system construction phase • One Design Class Diagram supporting all constructed use cases (an evolution of the Analysis Class Diagram) • Beta release software product • Updated Project development plan (based on use cases) • Detailed plan for System Deployment Phase (based on constructed and integrated product) For each use case constructed • An expanded essential use case (including a full set of alternative courses of events), • A real use case, • A design transition table, • Contracts, • Collaboration diagrams • Method specifications • An executable component that was integrated to form part of the final product • A complete and executed test plan • Management/owner/user satisfaction with the executable component
System Deployment	System Construction Phase output:- The System Deployment Plan, Beta software release, management approval and resource commitment	• Completed system installation, user training and user system acceptance • Project termination (subsequent enhancement/maintenance should be a separate project)
System In Use	System Deployment Phase output and management acceptance of project completion	• Significant changes or enhancements to the existing system are requested • Management approval for a new project to commence • Commitment of resources to undertake the System Investigation Phase

Table 1: Phases, Entry and Exit Criteria for the Prescriptive Simplified Method

Phase	**Entry Criteria**	**Exit Criteria**
Analysis	System Definition (& System Investigation) Phase output, plus management approval and resource commitment Preliminary System Construction Phase output Any previous UCDC output	• An expanded essential use case (including a full set of alternative courses of events) • An analysis class diagram (evolved to support the use case) • UCDC team satisfaction with the other exit criteria
Design	All previous software development method phase output plus management approval and resource commitment Preliminary System Construction Phase output Any previous UCDC output From this UCDC UCDC Analysis Phase output	• An enhanced expanded essential use case, • A real use case, • A design transition table, • System event contracts, • Collaboration diagrams • A design class diagram (evolved to support the use case) • Method specifications • UCDC team satisfaction with the other exit criteria
Construction	All previous software development method phase output plus management approval and resource commitment Preliminary System Construction Phase output Any previous UCDC output From this UCDC UCDC Design Phase output	• An executable component ready for testing • Source code for the component • UCDC team satisfaction with the other exit criteria
Testing	All previous software development method phase output plus management approval and resource commitment Preliminary System Construction Phase output Any previous UCDC output From this UCDC UCDC Design Phase output	• A complete and executed test plan • An executable component tested to use case level, ready for integration with other components at the same development stage • UCDC team satisfaction with the other exit criteria
Integration	All previous software development method phase output plus management approval and resource commitment Preliminary System Construction Phase output Any previous UCDC output From this UCDC UCDC Test Phase output	• An increment of the beta release product • Management/owner/user satisfaction with the integrated executable component

Table 2: Phases, Entry and Exit Criteria for the Use Case Development Cycle (Embedded in the System Construction Phase of the Prescriptive Simplified Object-Oriented Method)

Figure 2 - Standardised Description of PSM SDC and UCDC Phases

1. Phase

All phases start with an introduction that positions the phase in the life cycle.

1.1 Purpose of the Phase

A description of what work will be done, and why it will be done.

1.1.1 Entry Criteria

More detail about the entry criteria (as described in Table 1 and 2).

1.1.2 Exit Criteria

More detail about the exit criteria (as described in Table 1 and 2).

1.2 Responsibilities

A description of who is responsible for the work that will be done, and why it will be done be a particular role. Presented using the following four headings.

1.2.1 Project Leader

1.2.2 System Developer

1.2.3 Business Manager/User

1.2.4 Project Owner

1.3 Software Development Method Tasks

A prescription of each task, how (in very fine detail) the task is done and what will be produced. Presented using the following three headings for each task.

1.3.1 Task Name

1.3.1.1 Technique

1.3.1.2 Artefact

1.3.2 Task Name

… etc …

1.4 Work Product Quality and Testing

A description of the standard the work should be and when and how peer and management reviews take place.

1.5 Phase Deliverables

A description of the actual deliverable for the phase.

1.6 Phase Review

A description of the review process for the phase.

A task is described in section 1.3.1. PSM is a top-down approach to development. We emphasise in this section the detail and knowledge about the system that is being captured.

The technique to develop the artefact for each task is described in section 1.3.1.1. We go to some length to prescribe exactly how to do the task. The three pilot studies have shown that it is critical for the students to be able to independently repeat the task. Therefore, the technique must be of fine detail. We took full advantage of existing "best practice" to describe the techniques. For instance we used structured programming and pseudocode to write method specifications. The meticulous and accurate nature of system development is emphasised throughout the PSM. The technique also describes the testing and quality assurance that should be done while doing the task.

The artefact (section 1.3.1.2) is the outcome of the task, having followed the technique. It is the tangible evidence that the technique has been followed, including the testing and quality assurance elements within the technique.

Section 1.4 ("Work Product Quality and Testing") explicitly draws attention to the testing that should be done with peers and other project participants. Section 1.4 reminds the student that system development is a team effort and describes how the team tests and assures quality at the end of each phase.

Prior to PSM, our experience was that students had difficulty constructing a deliverable at the end of each phase. Section 1.5 describes the deliverable that needs to be presented to show that a phase has been completed. This section also clearly sets out what will signify the end of the phase, that is the production of a complete, accurate and meticulously done deliverable.

The final section deals with phase review. Again our experience with students prior to PSM, was that students tended to gather requirements, feverishly work through the project and only delivered to the client at the very end of the project. The students do not seek management review frequently. As in the previous two sections, this section draws the work of each phase together to a certain conclusion that the students present to the client.

THE TEACHING APPROACH

PSM is intended for instruction, therefore we have also developed a teaching approach to accompany the PSM. The approach is bottom-up. Our approach is to first engage deep learning about the skills of system development tasks with surface learning about the UCDC and SDC. We follow up with deep learning about the deliverables, the SDC and UCDC. The third encounter with PSM is an independent undertaking by the students of a non-critical project from a genuine enterprise. A non-critical project means the enterprise is not jeopardised in any way if the project should fail or not meet all requirements. The enterprise is genuine. It is outside the university.

The PSM states that the SI phase is usually done by experienced developers and often with the support of financial advisers, especially for the business case. When students learn about the PSM for the first time (as in stage two, first component of this research project) it is recommended that a complete set of SI phase entry and exit criteria be provided. Our experience prior to PSM was that the enormity and complexity of the project overwhelms the students when they are faced with having to meet the SI exit criteria. When students learn about the PSM for the second time (as in stage two, second component of this research project) a partial set of exit criteria is provided. The students write the problem, client, and project plan (given the restrictions of submission dates and university datelines this is a very structured task). They also review and complete the business functions. It is during stage three that the students are expected to meet all phase exit criteria independently.

The SDef phase also has a breakdown of the instruction to meet the exit criteria, similar to the SI phase described above. Students are eased into the tasks of this phase. When students learn about the PSM for the first time (as in stage two, first component of this research project) they write high level use cases, draw the use case and analysis class diagrams. When students learn about the PSM for the second time (as in stage two, second component of this research project) they rank use cases and devise the project development plan including the detailed plan of the SC phase. It is during stage three that the students seek management/owner/user agreement.

The SC phase consists of one or more increments of the product. An increment contains one or more UCDC. When students learn about the PSM for the first time (as in stage two, first component of this research project) they write an expanded essential use case (including a full set of alternative courses of events). They also complete a design transition table, write contracts and draw collaboration diagrams for all the system events in the use case, and draw the design class diagram. When students learn about the PSM for the second time (as in stage two, second component of this research project) they are expected to meet the exit criteria for the UCDC with assistance. The students work on the project in teams. Each team is assigned a different use case. The students are encouraged to integrate their use case with others though this is not enforced. (The more talented students are likely to achieve this.) It is during the third stage that students have to integrate the executable components of use cases and complete the SC phase.

The SDep phase is left till stage 3 of instruction. By this time the students are familiar with a work breakdown structure. This will enable them to not only incrementally develop the product but also to incrementally develop the user documentation.

Given the time constraints on university courses we believe the teaching approach is a sound method of instruction of systems development. We have used part of the teaching approach during the pilot studies. We will be testing the teaching approach during the second and third stages of the research study.

CONCLUSION

This paper has described both the PSM and a recommended teaching approach. PSM is distinctive from earlier use case approaches because the role of use cases is explicit and localised. PSM uses a traditional software development cycle and embeds use case development in the system construction phase. PSM has been used in three pilot studies that have lead to some refinements. The pilot studies support PSM as a viable means of transitioning lay people to novice system developers.

The teaching approach is bottom-up. That is, students begin by performing low-level tasks within the PSM framework, where higher level tasks have been done for them. Later stages of the research project will evaluate the efficacy of this bottom-up approach.

The definition of PSM and an accompanying teaching approach was merely the outcome of the first stage of our project. Our overall goal is not to simply produce yet another method, but to validate it by tracking the development of a cohort of students, from the very beginning of their training through to their early experiences in industry.

REFERENCES

Biggs, J. (1999) *Teaching for Quality Learning at University*. Society for Research into Higher Education and Open University, Buckingham, England.

Constantine, L. and Lockwood, L., (1999) *Software for Use,* Addison-Wesley, Reading, MA.

Hellens, L. von, Wong, S. and Orr, J., (2000) IT Skills Requirements: Perspectives from Industry, *Proceedings of the 11th Australasian Conference on Information Systems*, Queensland University of Technology, Brisbane, Australia.

Jacobsen, I., Christerson, M., Jonsson, P. and Övergaard, G., (1992) *Object-Oriented Software Engineering*, Addison-Wesley, Reading, MA.

Jayaranta, N. (1994) *Understanding and Evaluating Methodologies, NIMSAD: A Systemic Framework.* McGraw-Hill, Maidenhead.

Larmen, C. (1998) *Applying UML and Patterns,* Prentice Hall, NJ

COPYRIGHT