Teaching Information Systems Development via Process Variants

Wee-Kek Tan
Department of Information Systems
National University of Singapore
Singapore, Singapore 117417, Singapore
tanwk@comp.nus.edu.sg

Chuan-Hoo Tan
Department of Information Systems
City University of Hong Kong
Kowloon Tong, Kowloon, Hong Kong SAR
ch.tan@cityu.edu.hk

ABSTRACT

Acquiring the knowledge to assemble an integrated Information System (IS) development process that is tailored to the specific needs of a project has become increasingly important. It is therefore necessary for educators to impart to students this crucial skill. However, Situational Method Engineering (SME) is an inherently complex process that may not be suitable for students to apply in a classroom IS development project. SME is defined as the systematic creation of new methods from parts of existing methods, i.e., the method fragments, by taking into account the specific business situation of each IS development project. A less complex pedagogical approach is to teach students how to design an IS development process variant that incorporates the building blocks of various existing processes in order to leverage the advantages of each individual process. This paper first proposes a framework for teaching students the designing of process variants, followed by a preliminary empirical study conducted in a genuine classroom setting to determine whether the framework benefits students. Through the preliminary study, we discuss how the student IS development project teams had successfully applied our framework to design and use their own process variants. The initial observations obtained from the study also suggest that students who designed their own process variant appeared to consistently outperform those who did not, i.e., students which opted to use the traditional waterfall model.

Keywords: Information Systems Development, System Process Models, Curriculum Design

1. INTRODUCTION

The development of an information system (IS) has always been a complex process necessitating the use of methodological approaches such as the systems development life cycle, which is a systematic process of creating a system (Carroll, 2003; Necco et al., 1987). Hence, numerous methods have been proposed in the past two decades (Van Vliet and Pietron, 2006). The number of methods is now a colossal figure, thus creating the challenge of how to go about choosing the appropriate one(s) for a given IS development situation (Jeyaraj and Sauter, 2005; Wynekoop and Russo, 1995). A viable solution to this problem is to apply method engineering (Iivari et al., 2000-2001) to customize an appropriate IS development method based on the characteristics and needs of a project. Method engineering refers to the systematic creation of new methods from the parts of existing methods that are known as method fragments (Brinkkemper et al., 1998). Through the application of method engineering, a software firm could obtain greater process flexibility that leads to increased competitive performance (Nidumolu and Knotts, 1998).

Although having a well-designed development method could improve the quality of an IS (Necco et al., 1987), many IS graduates are not sufficiently well-trained with respect to using method engineering for effective system analysis, design and development (Kim et al., 2006). A significant cause of this pedagogy gap is largely attributable to the inherent complexity of method engineering. Thus, while some pedagogical attempts have been made to teach

some of the latest agile methods in many schools (Schneider and Johnston, 2003), there remains a significant educational need to train students in selecting and tailoring an IS development process (used interchangeably with method) suiting their specific needs.

This paper, hence, presents a novel approach that is built upon the designing of an IS development process variant (Gnatz et al., 2001; Henderson-Sellers and Serour, 2005; Song and Osterweil, 1998) to address the gap between the existing knowledge and skills of students and the foundation required for practicing method engineering. A process variant incorporates the building blocks of various existing processes in order to leverage the advantages of each individual process (Gnatz et al., 2001). From a terminological point of view, process variant designing is similar to method engineering. But from an operational point of view, process variant designing is less radical and consequently, implementing it is less complex. Moreover, process variant designing is more apt for use in classroom teaching as educators are better able to define the IS development processes and the specific parts of each process that meet two criteria: 1) applicability to their IS development projects; and 2) familiarity to their students. Essentially, students should be able to confidently master the basic concepts of method engineering through process variant designing.

The research objectives of this paper are twofold. First, a framework for designing an IS development process variant is conceptualized as a viable pedagogical tool for teaching students the prerequisite knowledge to practice method engineering. Second, a small, preliminary empirical investigation is conducted in a real classroom setting with undergraduate students taking an Enterprise System Development (ESD) course in our University to determine whether the framework is beneficial to students. Our observations suggest that it is possible for students to learn how to customize their own IS development process variants which are tailored to the unique requirements of their projects. More importantly, we find that students who used a tailored process variant appeared to perform better, i.e., obtained a better letter grade for the course, compared to the students who did not do so, but instead used the traditional waterfall model.

2. THEORETICAL BACKGROUND

2.1 Situational Method Engineering

Before the advent of method engineering, researchers had noted the importance of selecting the correct IS development methods in order to ensure that an IS could be developed at a lower cost and in a shorter period of time while concurrently meeting the needs of the users (Davis et al., 1988). Consequently, different techniques were developed to aid IS developers in the methods selection process. For instance, Alexander and Davis (1991) organized IS process models into a three-level hierarchical classification and then defined a total of twenty criteria together with a mathematical model for selecting the most appropriate one for a particular project. Despite these herculean efforts, predefined methods could either be too generic or contain parts which were incompatible with real projects' characteristics (Brinkkemper et al., 1998).

The Situational Method Engineering (SME) discipline has thus emerged as an approach to addressing the increasing demand for complex enterprise-level systems in organizations by taking into account the specific business situation of each software development project (Harmsen et al., 1994; Henderson-Sellers, 2003; Ralyté, 2002). SME is a method engineering paradigm that builds on the supposition that a method is conceived as not being a single intertwined and interdependent entity but rather as one incorporating a set of distinct fragments (Brinkkemper, 1996). A method is defined as a standard and systematic way in which a task is accomplished (Brinkkemper, 1996). Developers using the SME would select and combine existing parts of a method, i.e., the fragments and not the entire method itself, to form an integrated set of method fragments (Harmsen et al., 1994). Such an approach allows developers to formulate the unique development method that specifically caters to the business needs, and more importantly, the needs of a project.

Numerous techniques, models and theories can be found in the extant literature that address how SME may be implemented (Brinkkemper et al., 1998). For instance, the S4 theoretical model suggests that the situation factors and performance indicators of an IS development project collectively describe the criteria determining the success of the project, i.e., the project scenario, which is then used to determine the method fragments to be selected (Klooster et al., 1997). This theory synergizes the more traditional SME approach that considers only the project's situation factors (Slooten and Brinkkemper, 1993) together with the risk analysis and management approach that emphasizes the importance of including certain IS development activities in order to reduce the overall risk of failure (Charette, 1989). The risk management aspect is manifested in the S4 theory as the performance indicators, which help to identify when a project might be in danger of failure.

Even with the aid of the various theoretical models, the entire SME process, from method selection, to method construction and to tool adaption, is an inherently complex process necessitating the incorporation of an array of intricate method knowledge (Harmsen, 1997; Mirbel and Ralyté, 2006; Tolvanen, 1998). In fact, the metamodeling process itself that precedes the actual SME requires the method engineer to perform numerous tasks such as identification of the techniques in the methods, determination of the object type's properties, determination of relationships, and many others (Tolvanen, 1998). A deep understanding of the relevant business domains and processes of the protagonist organization is also a prerequisite to deciphering the particular situation for applying the SME. For these reasons, at least, we reason that it is not feasible to impart in students a sufficient working knowledge of SME without overwhelming their intellectual ability and interest.

It is thus hardly surprising that despite the increasing importance attached to the use of the correct method tailored to the specific needs of an IS development project, many educators have continued to focus on the definitions and usage of selected methods in their entirety. In particular, recent literature has mainly focused on the use of agile methods (see Schneider and Johnston, 2003, for instance). Even the notable few exceptions such as Lemmen et al. (1999), who examined the educational effects of SME, have

not been able to incorporate SME into an actual classroom IS development project and assess its effectiveness. Our present research endeavor attempts to address this knowledge gap by not utilizing SME itself but rather a different approach based on IS development process variant designing.

2.2 Designing of Information System Development Process Variant

The available IS development processes, such as the traditional waterfall model, the spiral model and the unified software development process, each has its own distinctive advantages and disadvantages (Gnatz et al., 2001). To this extent, the ability to combine the advantages and disadvantages of each existing process to construct a new integrated process variant tailored to a particular IS development project has been proposed using process patterns, i.e., individual activities or parts of a process (Ambler, 1998, 1999; Bergner et al., 1998; Carroll, 2003; Carroll et al., 2006). Process patterns can be thought of as being equivalent to method fragments in SME. Essentially, process patterns describe and document the various development activities in a structured, well-defined and modular manner that facilitates reuse within a process framework for integrating different process models (Gnatz et al., 2001). Within this framework, each process pattern prescribes the required activity to be performed, which, upon completion, leads to the generation of some work product, i.e., the tangible output, which can be modeled using notational symbols. For instance, Cameron (2002) put forth the Work Product Descriptions (WDP) framework that specifies how work products may be joined together in some temporal order to create an IS development process variant.

Although conceptually similar to the end state of SME (Firesmith and Henderson-Sellers, 2002), using process patterns within the process framework is less tedious (Becker et al., 2007). On the one hand, SME typically requires an elaborate set of procedures and representations to model the method fragments (Brinkkemper et al., 1998). Moreover, SME is closely guided by method assembly rules that are grounded on mathematical principles and these rules are often expressed in complex, first order predicate logics (Brinkkemper et al., 1998; Serour and Henderson-Sellers, 2004). On the other hand, a process-based approach uses textual description to narrate factual information about the method fragments (Gnatz et al., 2001). Process variant assembly is based on contextual requirements of the particular IS development project such as the need to perform a certain activity in order to generate a required output artifact (Gnatz et al., 2001). The method fragments for each contextual requirement are then joined together based on their temporal precedence into the final completed process variant (Noll, 2003).

In the same vein, Song and Osterweil (1998) applied process programming to customize a precise software development process for specific development projects. Using the Object Modeling Technique (Rumbaugh et al., 1991) and the APPL/A process coding language (Sutton et al., 1990), the authors successfully tailored the Booch Object Oriented Design Method (Booch, 1991) for different project requirements based on various project properties such as the type of programming language, required documentation and experience of the development team.

Beyond project properties, the features of the IS, i.e., "the coherent and identifiable bundle of system functionality that helps characterize the system from the user perspective" (Turner et al., 1999, pp. 3), can also play an important role in the development process (Turner et al., 1999). Specifically, features can be thought of as forming a bridge linking the problem domain to the solution domain. Each feature represents a logical module of user requirements that is represented by various process artifacts, e.g. class diagrams and test cases. Collectively, these artifacts realize the system design for implementing a particular feature in the solution domain.

Additionally, features also impact upon several process activities such as requirement engineering, system design and architecture, and testing. As a concrete example, Van Gurp et al. (2001) used features to represent variability in the development of software product lines and proposed a framework based on how design decisions in the development process can be affected by software features.

3. FRAMEWORK FOR DESIGNING AN INFORMATION SYSTEM DEVELOPMENT PROCESS VARIANT

In this section, we present a framework for IS development process variant designing that is contingent on the properties and features of the specific IS development project. The general thesis underlying both SME and process variant designing is such that the derivation of a set of methods for a project can be achieved in three steps: 1) understanding and defining the project situation (i.e., in our case, students were asked to consider the project properties and features) as well as the method requirements, 2) selecting the method fragments that fulfill the requirements of the project (Aydin et al., 2005), and 3) assembling the method fragments in order to form an integrated method (Ralyté, 2002; Tolvanen, 1998). While this approach resembles the software development process approach in Motorola (Fitzgerald et al., 2003; Fitzgerald et al., 2006), our pedagogical objective was not SME itself but rather more closely aligned to a simpler case of designing an integrated process variant from existing processes that the students had previously learnt. Our own approach towards IS development process variant designing is depicted in Figure 1. This approach is grounded on the same three steps used in SME but without much of the complexity involved.

3.1 Project Situation Definition

Step 1 of SME typically involves the consideration of numerous situation-independent and situation-dependent criteria (Tolvanen, 1998). The former include generic criteria independent of a specific IS development situation such as ease of use and ease of learning. The latter criteria focus on choosing method fragments that are most appropriate for a specific IS development situation and include factors such as the target organization's hierarchical structure. In our process variant designing framework, we only consider two situation-dependent criteria, namely project properties (Song and Osterweil, 1998) and system features (Turner et al., 1999). These two criteria are deemed to be more salient to the students when they analyze the project specifications, which document the business requirements of the protagonist organization.

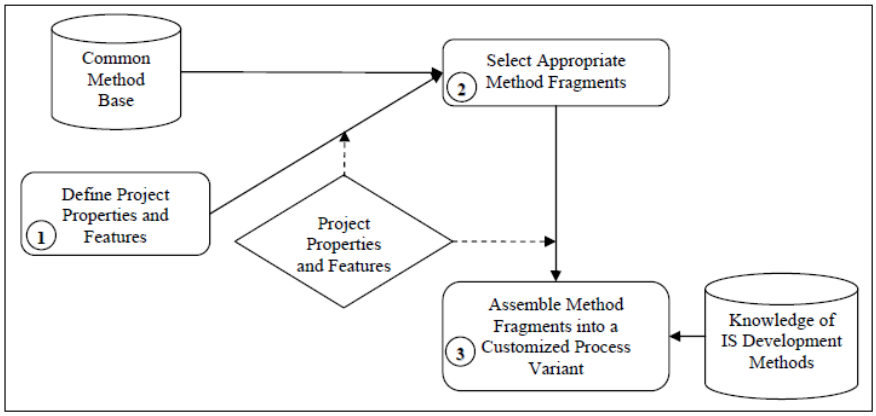

Figure 1. Framework for Designing an IS Development Process Variant

The project properties considered in our framework include the degree of difficulty, familiarity, complexity and scale of the system to be developed (Song and Osterweil, 1998). In addition, a crucial and major project property taken into consideration was the degree of coupling between the system's functionalities. This property refers to whether the system functionalities are loosely or tightly interrelated or interdependent on each other for the completion of a business process or service. In particular, functionalities that support business processes cutting across functional or departmental boundaries can be expected to be closely coupled with other related functionalities. This property is also important because it is closely related to the features of the system which are deemed to be at a higher level of encapsulation for related system functionalities (Turner et al., 1999). From the features perspective, our framework focuses on the feature breadth versus feature depth of the particular IS to be developed. An IS with multiple but simple features can be considered as possessing high feature breadth but low feature depth, whereas another one with fewer features but that is of higher complexity can be considered as possessing low feature breadth but high feature depth.

Collectively, the two situation-dependent criteria in our framework are closely related to the two sources of complexity that are commonly associated with IS (Iivari and Koskela, 1987). The scope of the application domain, e.g. the number of transaction types processed by the IS, is mapped to the feature breadth versus depth dimension. The inherent multidimensionality of the IS involving an interaction between technical requirements, organizational structure requirements and social communication requirements is mapped to the functional coupling property. The feature criterion and the functional coupling property also have a major impact on the size and scope of the IS, which would in turn affect other project properties.

3.2 Method Fragments Selection

Step 2 of SME involves the selection of appropriate method fragments classified along many dimensions such as perspective, abstraction and layer of granularity (Brinkkemper et al., 1998). Our process variant designing framework focuses only on the perspective dimension, which classifies method fragments along the two sub-dimensions of process, i.e., the task to be executed, and the product, i.e., the deliverables upon completion of the task. The perspective dimension is easier for the students to grasp since they would have learnt the theoretical knowledge of each method fragment and actually created some of the associated products in the prerequisite courses., The required method fragments would be selected, depending on the situation-dependent criteria of the IS to be developed. To illustrate, a small development team working on an IS for a familiar industry with few features of low functional depth would only need to undergo a simple business analysis rather than complex process modeling. The former only involves speaking to the end business users to gather their requirements whereas the latter would require a lengthy period of modeling various business processes in the organization.

Additionally, our process variant designing framework restricted the method fragments to those compatible with the object oriented analysis, design and programming paradigms in order to be consistent with the use of the object oriented Java language. Our framework also removes redundant method fragments that performed similar tasks (Fitzgerald et al., 2006). These techniques reduce the ambiguity and conflict involved in the method fragments selection. For instance, it is illogical to assemble a traditional data flow diagram with an Unified Modeling Language (UML) activity diagram since the former is for structured system analysis and design whereas the latter is for object oriented system analysis and design. Moreover, both method fragments are essentially performing the same task of system process modeling, and selecting both fragments would result in redundancy.

In conjunction with our framework, a method base of carefully chosen method fragments was created. One of the key criteria for choosing the method fragments was that students needed to possess the theoretical knowledge, acquired from prior prerequisite courses, to apply them. The method fragments were organized in the method base along the perspective dimension, i.e., in terms of process and product. The method base is shown in Table 1.

3.3 Process Variant Assembly

Step 3 of SME typically involves complex assembly rules (Brinkkemper et al., 1998) and our process variant designing framework opts to skip these rules in favor of the temporal precedence heuristic (Gnatz et al., 2001) that is more manageable for the students to adopt. Briefly, our framework

Development Activity	Method Fragment	
	Process	**Product**
Project Management	**Project Management:** Resource allocation and scheduling with setting of milestones.	Gantt Chart and Resource Usage.
Business Analysis	**Requirement Analysis:** Understanding the business and functional requirements.	Requirement specification report with list of assumptions.
	Process Modeling: Business domain analysis and in-depth organizational business process analysis.	UML activity diagram and process descriptions.
System Analysis	**Use Case Modeling:** Structured view of system functionality using OOAD and UML.	Use case diagrams and descriptions.
	Domain Modeling: Construct initial model of real-world system using OOAD and UML.	Class diagrams (without attributes and methods).
System Design (Logical)	**Class Modeling:** Construct complete model of real-world system using OOAD and UML.	Class diagrams (with attributes and methods).
	Interaction Modeling: Model component interaction using OOAD and UML.	Sequence and collaboration diagrams.
	Screen Flow Modeling: Model the screen flow using structured walk-through of system from user view using agile modeling.	User interface flow diagrams or storyboard.
System Design (Physical)	**State Behavior Modeling:** Model objects state-dependent behavior using OOAD and UML.	Statechart diagrams.
	System Architecture Design: Modeling components and sub-components division, connection, interaction and interfacing.	System architecture diagrams.
	Data Modeling: Model the data requirements of system using the relational model.	Entity relationship diagrams and data dictionary.
	User Interface Design: Design the graphical user interfaces using the User Centered Design (UCD) philosophy.	User interface prototypes.
	Algorithm Design: Design the various algorithms for performance of computationally complex tasks.	Pseudo codes of algorithms.
Implementa-tion	**Prototyping:** System development broken down into smaller logical groups of related functionalities and developed in phased increments.	Working prototypes of system for users' evaluation.
	Risk Analysis: Risks are explicitly assessed and resolved throughout the process.	Risk analysis reports.
	Testing: Develop and execute unit and integration testing strategies and plans.	Test plans and results.
Testing	**User Acceptance Testing:** Testing against specifications and system walk through with user versus requirement specifications.	Test plans and results.

Table 1. Method Base Used in the Framework

Method	Description
Traditional Waterfall	Sequential development process for unambiguous and well understood user requirements where the business environment is relatively stable. Suitable for projects that require little change. Possible to move back to previous stage, 1 step at a time.
Incremental Waterfall	Entire system is broken down into multiple increments with each increment delivering part of the system features. The development of each increment is a mini-waterfall. However, all user requirements are fixed before the start of the first increment and no change can be made subsequently.
General Prototyping Process	Identify problem and gather initial problem to develop prototype. Implement and use the prototype. If prototype is efficient and effective, transit to operational system. Otherwise reanalyze the problem and make revisions to enhance the next prototype.
Rapid Application Development	Iterative and incremental development process focusing on developing prototype (extreme prototyping). Use concurrently with various techniques to speed up system development.
Spiral Model	Evolutionary (iterative) approach of system development in which the same set of prescribed activities is repeated over a number of cycles. Basically incorporates design and prototyping work in stages instead of at the beginning of the development.
Rational Unified Process	Iterative software development process framework that is highly adaptive. Project teams can tailor the framework to select the elements of the process that are appropriate for their needs. In other words, they are not bound to a single pattern of their developing system.
General Agile Methodology	Focus on a small set of software development principles such as being adaptive to changes in user requirements, small team development, rapid and multiple development iteration, to deliver system in an incremental fashion.
eXtreme Programming	One of the most common forms of agile methodology that emphasizes collocation of a development team with end users, informal solicitation of user requirements using story cards and multiple iteration of analysis, design, coding, testing and integration, using pair programming. Each cycle cumulates in the release of a working system.

Table 2. IS Development Methods Used in the Framework

teaches students to assemble the method fragments in the order of the system development activities that should be executed (Noll, 2003) with respect to well known methods learnt in the prerequisite courses.

The complete list of methods used in our framework is shown in Table 2. Taking the traditional waterfall model as a reference point, the students should then assemble method fragments relating to analysis, design, coding and testing in that order. While it is plausible to skip some steps or insert additional ones that are relevant to the particular IS development situation, it would not make sense to assemble design-related fragments before analysis-related fragments.

Emphasis was also placed on the appropriate use of iterations and increments that are commonly found in modern IS development processes such as incremental waterfall, general prototyping process and the spiral model (Aydin et al., 2005). Nevertheless, the assembly process is closely guided by the two situation-dependent criteria of the IS development project. As an illustration, a simple IS can be developed in a single iteration whereas a complex IS would require multiple iterations of prototyping and testing. Moreover, a complex and unfamiliar project would require a dedicated risk management process, possibly at the end of each iteration.

4. RESEARCH METHODOLOGY

4.1 Research Design

To determine whether our framework is indeed beneficial to students, we conducted a preliminary empirical study in a natural classroom setting (Neuman, 2006). Undergraduate students taking an ESD course in our University over a 13-week semester were taught the framework and subsequently applied process variant designing in their project. We next analyzed the process variants designed by the students as well as the post-hoc performance, i.e., the final letter grade obtained, to assess whether the students had benefited from our framework. The main advantage of this approach was that all the students were taught the same approach and thus none was disadvantaged in the assessment. Moreover, the authors together with the teaching teams had the opportunity to observe throughout the semester how the students had applied the process variant designing framework.

4.2 Course Structure

ESD is an advanced course offered in our IS bachelor program. Students taking the degree are required to pass the course, i.e., obtain a minimum letter grade of D, and typically enroll for ESD in their 3rd-year of study. The objective of the course is to offer a suitable avenue for the students to apply what they have learnt in prior introductory courses taken at a cursory level, which include the fundamentals of software engineering, system analysis and design, database systems, and Java programming. Students taking ESD are therefore expected to possess adequate programming skills and related software engineering knowledge. These include common development methods such as the traditional waterfall model, incremental waterfall model and the spiral model. This course allows the students to experience, for the first time, a simulated IS development environment with a sufficiently high degree of realism, i.e., the business context is written by referring to market practices and business operations. In addition, they are exposed to potential group dynamics and conflicts (in the event that they should arise).

Considering the complexity of fulfilling the course requirements, students are awarded two times the number of credits than those awarded to the usual courses the University offers. The course presents an opportunity for the University to assess the capacity of the students to build an enterprise-level IS within a semester of 13 weeks. During the first week of the semester, students would form teams of five to six and the project specification would be distributed during the first lecture. Teams meet formally with their project advisors, who act as system users, for an hour each week to clarify the system requirements and to report on their progress. The scope of the project normally requires teams to meet outside of regular classes and consultation times. The teams are required to submit an initial system proposal in Week 03 and a final system analysis and design report in Week 11. There are altogether three incremental system releases that the teams must deliver: 1) first system release in Week 07; 2) second system release in Week 10; and 3) the final system release in Week 13.

In order to inject realism and to minimize copying of work across semesters, we changed the project specifications, i.e., the business domain and the associated IS to be developed, every semester. This approach actually complements the pedagogical objective of our process variant designing framework since it would compel the students to design the best process variant that would cater to the different system development needs. The selection and drafting of the project specification are executed rigorously. The choice of the system depends on 1) the complexity of the analysis, 2) the current market demand, and 3) the feasibility of developing the system within 13 weeks. Inputs and comments from the industry with respect to the system to be developed are highly sought after. Such an attempt enables the course to be better aligned to the development and practice of industry in general.

4.3 Overview of Study and Subjects

The study was conducted in the semester which ran between January and May, 2008. The project specification requested the students to develop an Integrated Resort Management System (IRMS) to support the various business operations of a large scale resort. Furthermore, students were required to implement the standard administrative functions typical in a large-scale organization and factor in adequate security controls. To encourage the creation of variability from a single project specification, the students were allowed to "reconfigure" the required modules based on their own business assumptions which were subjected to approval by their respective project advisors. Specifically, the business operations were split into four main business areas, namely hotel, casino, convention center and shopping mall. The functional requirements were divided at a finer granularity into 10 core feature modules consisting of security, hotel room, casino, shopping mall tenant, entertainment, convention center, banquet, event, employee and automatic alerting system.

The students were told that the choice of feature breadth, (i.e., the number and variety of features,) and depth, (i.e., the complexity of the features), was entirely theirs as long as the basic features were included. The emphasis was on analyzing, designing, and implementing an integrated system coherent with the organizational business objectives. For this purpose, the students were presented with the case situation that emphasized the delivery of a reliable high-end hotel and resort management system specifically tailored to the unique requirements of the client. Students were also specifically told that they would be assessed on their documentation, project management, graphical user interfaces and other soft skills such as presentation and minute-taking during consultations.

The descriptive statistics of the students taking the ESD course are listed in Table 3. Altogether, 151 students took the course and they were grouped into 25 teams of between five to seven students per team with the average team-size being 6.04 ($\delta = 0.351$). The gender proportion of the students reflected the general demographic trend of IS majors in our University, with more males than females. The majority of the students were 3rd year undergraduates.

Item	Breakdown	Statistics
Gender of Students	Male	87 (57.62%)
	Female	64 (42.38%)
Year of Study	3rd Year	128 (84.77%)
	4th Year	23 (15.23%)

Table 3. Descriptive Statistics of Students

4.4 Research Procedure

The IS development process variant designing framework was taught to the students during a series of pre-semester workshops along with other materials pertaining to IS development using the chosen development platform, i.e., Java Platform, Enterprise Edition (Java EE). The importance of designing and utilizing an appropriate process variant in aiding their project work to develop an effective IS was emphasized to the students during the workshops. In brief, the students were taught to begin by defining the properties and features of the system that they proposed to develop. Based on the unique properties and features, each team then proceeded to select the appropriate method fragments from the common method base shown in Table 1, and to assemble them into their own customized process variant.

The knowledge on process variant designing was put into practical application during the first two weeks of the semester when the students went about designing their own process variants for developing their respective IRMS. This was done concurrently with the other project tasks. The students received further guidance from their respective project advisors during the consultation sessions in walking through each of the three steps of the designing process. The students were required at the end of Week 02 to explain the design rationale for their team's respective process variant as well as its feasibility, given the scope of the project, i.e., the features that the students proposed to be implemented. The process variant was not cast in stone and students were encouraged to make modifications along the way. Teams were expected to finalize their process variant by Week 03 and document the process variant in the system proposal.

No specific tools were adopted, unlike in SME where a Computer-aided Methods Engineering (CAME) tool is typically used (Tolvanen, 1998). This was intended to reduce the associated learning curve. Instead, the students were instructed to develop their own process variant using standard word-processing software that supported flowchart drawing. In other words, the final process variant was represented using the standard flowchart notations that the students were familiar with. The correctness of the process variant was then verified by the project advisors.

The results of the study are discussed in the following two sections. We first present a qualitative analysis of the process variants designed by the students to illustrate how our framework was applied by the students. This is followed by an initial quantitative assessment of the viability of process variant designing based on our framework in aiding students in their IS development endeavor.

5. PROCESS VARIANT DESIGN IN ACTION

Based on our observations, the students generally designed their process variants contingent on whether they had decided to develop a system that supported numerous business features but with fewer complexities or details, compared to one that supported fewer but more advanced business features. Consequently, the functional coupling property and feature depth/breadth criteria were used as the primary dimensions to classify the process variants designed by the students. The following discussion on the different types of process variants designed by the students will be structured along these two dimensions.

The process variants designed by the students could be broadly classified into three categories: 1) breadth and loosely coupled, 2) depth and tightly coupled, and 3) breadth plus depth and moderately coupled. Furthermore, while the process variants developed for each category were not entirely similar, they closely resembled one another and could be collapsed into a single generic variant for each category to facilitate our discussion. Another interesting observation was that some of the teams opted to fall back on the traditional waterfall model regardless of the complexities of their proposed systems. The implications of this observation will be discussed in Section 6.

5.1 Breadth and Loosely Coupled

This category involved a proposed system that provided only basic features for each of the 10 modules in the specifications. Such a system was designed to provide the bare minimum support for the four business areas of the resort and there were few interactions between each area. One of the evaluators commented that although this category of systems was technically easier to implement and thus unlikely to suffer from technical errors; the system however viewed the resort as functional silos and suffered from poor creativity that limited future expansion of the resort's business processes and services.

Most of the teams with projects of this nature opted to assemble the method fragments into an incremental waterfall-like process variant shown in Figure 2. The common reason given was that such a process variant allowed them to "break down" the system into three increments consistent with the two prototypes and the final deliverable mandated by the course requirements. Since the functional coupling was minimal, the teams could then deliver parts of the required features with each increment. This was also the main reason teams using this process variant did not perform process modeling compared to teams in the latter two categories. Some of the teams split the increments bearing in mind the priority of feature modules, i.e. releasing the more important modules in their entirety first. The remaining teams tackled all feature modules concurrently but focused on the more important sub-tasks within each module. Either way, the process variant shown in Figure 2 provided the flexibility for the teams to work on the feature breadth of the system. However, using this process variant prevented the teams from making changes to the features in the later increments since the requirements were fixed at the beginning of the project.

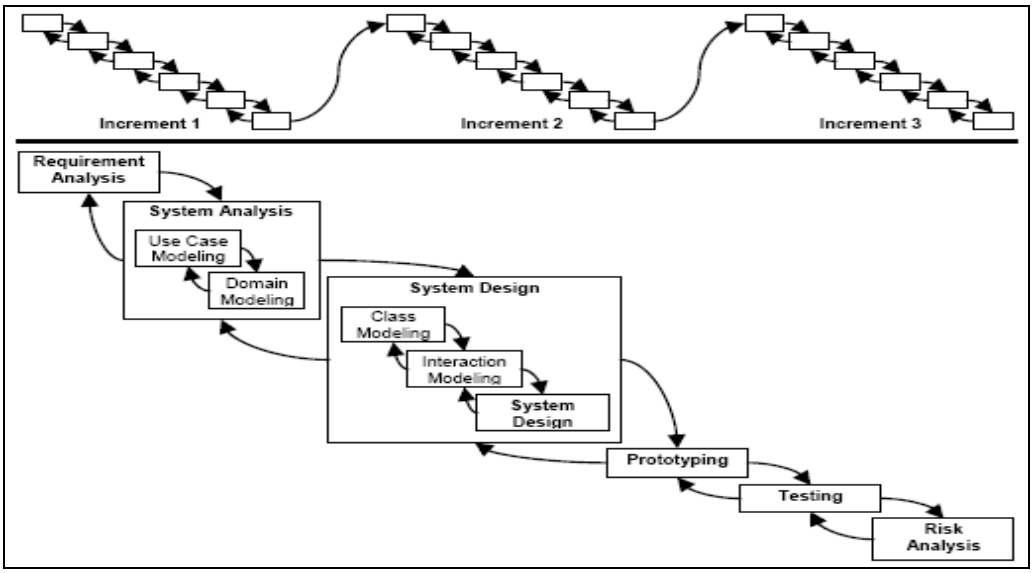

Figure 2. Generic Version of the Process Variant Adopted by Students Classified Under Breadth and Loosely Coupled System

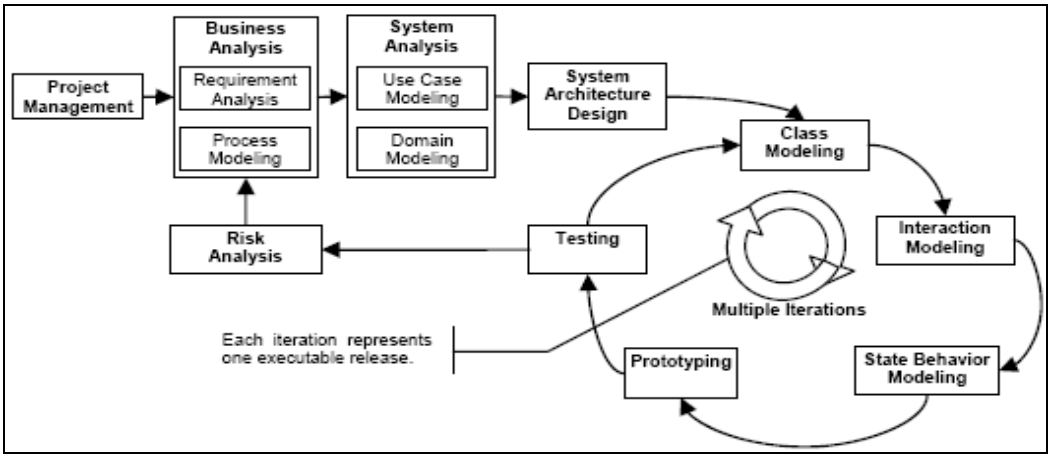

Figure 3. Generic Version of the Process Variant Adopted by Students Classified under Depth and Tightly Coupled System

5.2 Depth and Tightly Coupled

The proposed systems under this category are meant to focus on one particular business area of the resort and develop advanced features around it. The remaining business areas and modules were given basic to moderate attention. The general challenges faced by the teams included: 1) the need for the system architecture to be sound, 2) the need for team members to have a detailed understanding of each other's work, and 3) the requiring of intensive coding for the advanced features. Consequently, the process variants assembled by the teams (see Figure 3) closely resembled those of the programmer-centric Extreme Programming (XP) method which focuses primarily on the programming aspects of system development (Beck, 2000; Schuh, 2005).

The programming iterations were relatively short, each lasting about 1-3 weeks with the norm being between two weeks. Each of the iterations involved system design, intensive programming and testing. Based on observations from the project advisors, each team typically sub-divided themselves into various small groups of 1 or 2. For instance, once the team had finished the system designing, 2-3 persons were assigned to the programming work while the remaining team members were responsible for formal documentation and development of test cases. Each of the short iterations cumulated into an executable release which was put through the required integration testing to ensure compatibility with the previous releases. The team members then conducted a risk analysis before proceeding with the next iteration. The number of iterations was typically greater than the mandated three system releases as the teams preferred to break the programming tasks into more manageable portions.

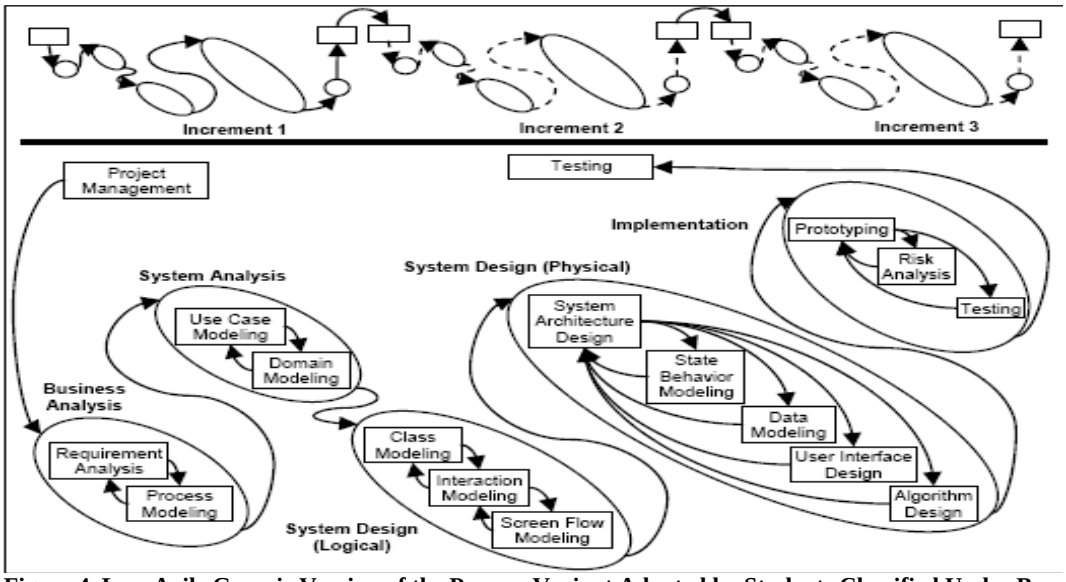

Figure 4. Less Agile Generic Version of the Process Variant Adopted by Students Classified Under Breadth Plus Depth and Moderately Coupled System

5.3 Breadth plus Depth and Moderately Coupled

This category included proposed systems that attempted to implement relatively advanced and interlinked features for almost all of the business areas and modules. These teams typically had many long discussion sessions with their project advisors on how to implement a comprehensive system capable of supporting business processes and services that cut across different business areas or functional departments of the resort. All the evaluators generally reached a consensus that this category of systems was the most technically and logically demanding of the three categories. Teams that developed systems in this category generally faced problems from both the first two categories in addition to a more complex set of business requirements. Thus, we did not find it surprising that the teams developed process variants that were slightly similar to the first two categories. We further noticed that the process variants developed by the teams could be sub-divided into less agile and more agile variants. By more agile, we refer to a closer resemblance to one or more of the agile methods (see Schuh, 2005, for example).

We will first discuss the less agile process variant, (see Figure 4), which resembled a heavily redesigned adaptation of the incremental waterfall model. By and large, the teams adopting the less agile approach assembled their method fragments into a coherent process variant that exhibited two characteristics: 1) it was a hybrid of the traditional waterfall model and the incremental waterfall model; and 2) it comprised three increments (i.e., iterative loops) corresponding to two intermediate prototypes and the final system. Like the breadth and loosely coupled category, this approach enabled the project advisor to monitor the progress of each team and assess their continual efforts, thus discouraging teams from deferring developmental work to the last weeks of the semester that would often result in a rushed job. Hence, this approach was still in accordance with agile software development in which the emphasis is on early and continuous delivery of the software product. However, the lower agility was a consequence of the inherent weakness in the strict incremental waterfall model which compelled students to execute all the intermediate methods thus plausibly exhausting precious time, given the tight dateline. To compensate for this weakness, some of the teams incorporated optional method fragments into their process variants. The dashed arrow connectors in increments 2 and 3 of Figure 4 indicate that the component intermediate method fragments are optional. In other words, students might skip certain method fragments if the particular prototype revision did not require changes to be made to them.

Whenever possible, students also tried to include concurrent execution of method fragments to enable optimal utilization of the limited human resources. During the system design (physical) phase, the teams were encouraged by the project advisors to sub-divide themselves into pairs based on the expertise of individual members to work on separate method fragments. The same concurrent execution could be found in the implementation phase. In a strictly incremental waterfall model, students would be compelled to execute all the method fragments sequentially. Another interesting observation is that students attempted to incorporate all of the method fragments embedded in the method base into their process variant. A plausible explanation was that given the substantial challenges posed by the feature breadth and depth and the intricacies of the businesses processes, students might have felt more confident following a more rigorous albeit a more tedious process variant.

The more agile process variant (see Figure 5) resembled the XP iteration model in the depth and tightly coupled category. However, three key distinctions made it more useful for teams that attempted systems with both breadth and depth together with moderate functional coupling. First, this process variant featured more method fragments resulting in a more rigorous process variant compared to that presented in Figure 3. This is similar to the rationale of the less agile process variant in Figure 4. However, the current process variant gained agility because of the concurrent method fragments for performing screen flow modeling, user

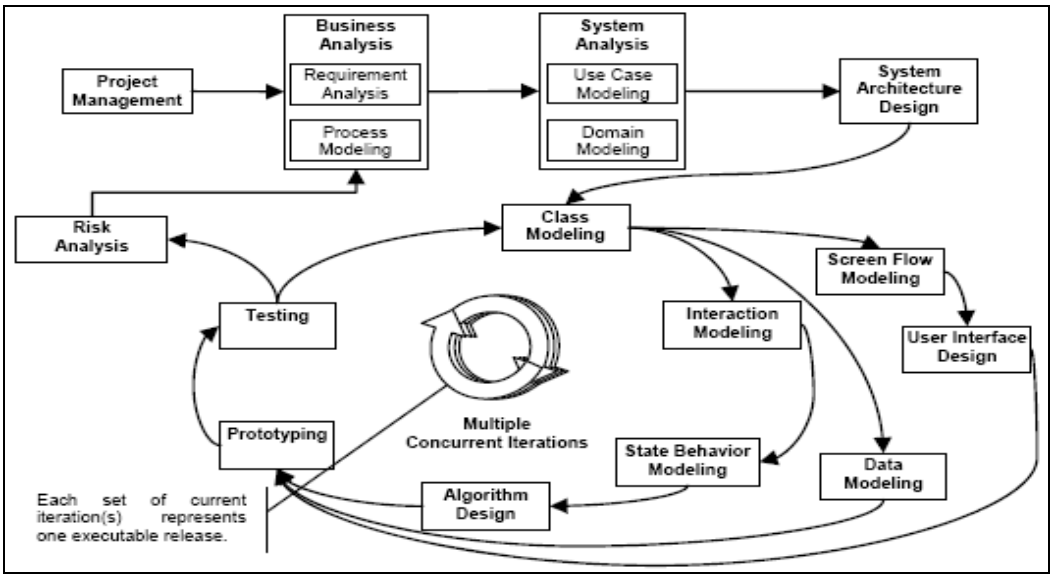

Figure 5. More Agile Generic Version of the Process Variant Adopted by Students Classified Under Breadth Plus Depth and Moderately Coupled System

interface design and data modeling, which was the second distinction. Third and most importantly, the current process variant allowed concurrent programming iterations along the lines of the Feature-Driven Development (FDD) agile methodology (Schuh, 2005). In other words, teams were better able to sub-divide their members into mini-development groups to tackle tasks of a finer granularity than the complete prototype release catered for by the process variant shown in Figure 4. For instance, each group of students could undergo multiple iterations in delivering the features for a single prototype. This facilitated a more effective integration with features created by other groups.

The fact that the students had designed and applied three different categories of process variants was a clear indication that our framework is a viable pedagogical tool for aiding students in learning IS development process variant designing. Given the close similarity between process variant designing and SME, we believe that process variant designing, being the easier technique to apply, could serve as a useful starting point for students to acquire the knowledge and practical experience to eventually practice SME. A logical next step is to determine whether process variant designing indeed leads to better IS development performance, i.e., whether process variant designing benefits students.

6. ASSESSING THE BENEFIT OF PROCESS VARIANT DESIGNING FOR STUDENTS

The authors and the teaching team generally agreed that it was difficult to assess the effectiveness of the various process variants developed by the students. Consequently, the task of determining whether process variant designing benefited the students is a non-trivial one. This was because the grading of the project systems was based on a checklist that emphasized the match between proposed business requirements and actual implemented system features as well as the level of complexity, integration and technical errors. The appropriate design and use of the IS development process variant was not an explicit assessment criterion since this was not the main objective of the ESD course. Thus, it was entirely possible that a team using the traditional waterfall model could deliver a better system than a team using any of the four generic process variants discussed previously. In fact, this was indeed the case with one team that used the traditional waterfall model obtaining a B+ grade.

Nonetheless, we deduced that if the teams had designed an appropriate process variant tailored to their specific needs, it should aid them in the development process, thus leading to the delivery of an overall better system compared to those who simply used the traditional waterfall model. Thus the final letter grade awarded to the teams was used as a surrogate measure of relative effectiveness. This approach provided us with initial objective figures to ascertain the benefit of process variant designing for the students. The fact that there were teams who opted to use the traditional waterfall model provided a control group for comparing against those teams that applied process variant designing. This mitigated the weakness of the non-experimental design adopted for the study. The grading criteria used are shown in Table 4. We used the final letter grade that included the documentation assessment instead of solely evaluating the development aspect because documentation is an inseparable part of IS development. In fact, the output of most method fragments collectively constituted the bulk of the contents of the report.

In addition, during the dialog sessions held at the end of the semester, the students were asked to rate individually their perception of the usefulness of the IS development process variant designing framework that they had used for the project (using a 7-point scale). Presumably, students who designed their own process variant should find them more useful compared to the traditional waterfall model. We also solicited feedback from the students which were tape-recorded during the dialog session. The summary of the findings are discussed in the following paragraphs.

The final team grade was translated into the respective grade point on a 5-point scale with A+/A being equivalent to

Assessment Component	Weight	Breakdown
System Proposal	10%	N.A.
System Analysis and Design Report	20%	
First System Release	10%	Conformance to Business Requirements and System Proposal = 30% Feature Breadth = 30% Feature Depth = 40%
Second System Release	10%	
Final System Release	50%	

Table 4. Course Assessment Criteria

5.0 and each subsequent half-grade step on a decreasing interval of a 0.5 grade point. The lowest team grade awarded was D+ and no team failed the course. The cross-tabulation of the final team grade against the process variant categories is shown in Table 5. The mean grade point of teams who used the traditional waterfall model was 2.625 (δ = 1.0308) and lower compared to those that designed their own process variants, which was 3.762 (δ = 0.1746). An independent-samples t test however indicated that the mean difference only approached but did not reach statistical significance, assuming unequal variance (ΔMean = -1.137, t = -2.089, p = 0.110). This was most likely due to the smaller number of teams who used the traditional waterfall model and the fact that one of them obtained a good grade. Although teams that designed their own process variant generally obtained a better grade point compared to those that did not, this inference did not reach statistical significance.

Among the four different categories of process variants, there was no significant difference in the mean grade point. This was indicated by a one-way analysis of variance (ANOVA) test (F (3, 17) = 0.382, p = 0.767). Thus, teams that designed their own process variants obtained comparably grade points. Interpreting these two findings together, we conclude teams that designed their own process variants appeared to consistently outperform teams that did not, although caution should be exercised in drawing this conclusion due to the non-significance of the first finding.

It was entirely possible that the better performance obtained by teams who designed their own process variants was attributable to their inherent capabilities instead of the actual process variant used. In the absence of any suitable measures on the students' capabilities, we analyzed the amount of development effort expended by each team with respect to the final letter grade obtained. Development effort was operationalized as the mean man hours reported by each team for the entire project. This figure was calculated using the total man hours expended by each team member as reported in the Gantt chart of the system analysis and design report. It is imperative to note that these are reported man hours and teams could expend more than this amount of time on the project on their own. We assume that the students worked on the project only during the weekdays and expended a maximum of 6 hours per day. This takes into consideration the normal lesson hours and the possibility that the students needed to prepare for other courses. Throughout the entire 13-week semester, the maximum man hour a student could expend was thus 390 man hours. However, the actual reported man hours ranged from a low of 205 man hours to a high of 405 man hours with a mean of 292.53 (δ = 40.058) man hours. The mean man hours expended for each team is shown in Table 6.

A two-way ANOVA test on the mean man hours expended by each team with the category of process variant and the final letter grade as the independent variables was conducted. The results indicated that there was no significant difference in the mean man hours across the different categories of process variant used (F (4, 5) = 1.932, p = 0.244) and the final letter grades obtained (F (6, 5) = 3.285, p = 0.106). The interaction effect was also not significant (F (9, 5) = 2.471, p = 0.166). We may therefore conclude that the final letter grade obtained by the team was not affected by the development effort expended by each team.

The mean rating of process usefulness among students who used the traditional waterfall model was 4.320 (δ = 1.3760) and lower compared to those that designed their own process variants, which was 5.120 (δ = 1.177). An independent samples t test indicated that the mean difference was statistically significant, assuming unequal variance (ΔMean = -0.799, t = -2.713, p = 0.011). Furthermore, among the four different categories of process variants, there was no significant difference in the mean rating (F (2, 123) = 0.306, p = 0.737). Thus, students who designed their own process variants consistently found it to be more useful compared to those who merely used the traditional waterfall model.

Moreover, through the verbal discussion held during the dialog session, we believe that imparting in students the ability to design their own process variants was generally effective in improving their performance in the course. On the one hand, teams that managed to complete their proposed systems in a planned, orderly fashion generally satisfied three criteria. First, these teams mentioned that they were able to articulate the specific situations of their project and develop a coherent business case as the basis for subsequent system development activities. Second, these teams told us that they were able to assemble together a set of method fragments that they perceived to be suitable for the specific variation of their proposed system which was identified earlier. Lastly, most of the 21 teams that designed their own process variant were able to execute diligently the process variant as they had planned it, throughout the entire 13-week semester. It should however be noted that, based on the feedback from the project advisors, there were indeed a few teams that required their project advisors to exercise additional supervision in order to follow through with their respective process variants.

On the other hand, the handful of 4 teams that opted to follow the traditional waterfall model commented that they had tended to fall behind project milestones because of the inability to overcome the scale and complexity of the project specification. Furthermore, most of these teams admitted that they had had to scale down the feature levels of their final delivered systems due to various intermediate programming difficulties and integration problems.

Generally, the teams that opted to spend more time during the first two weeks on developing their process variants appeared to be rewarded with a systematic IS development process variant tailored to their specific needs. This aided them in their subsequent development activities, including the facilitation of short programming iterations,

Process Variants	Final Team Grades								
	A+/A	A-	B+	B	B-	C+	C	D+	Total
Traditional Waterfall Model (Did not design own process variant)	0	0	1	0	0	2	0	1	4 (16%)
Breadth and Loosely Coupled	0	1	1	2	1	1	0	0	6 (24%)
Depth and Tightly Coupled	2	1	0	2	2	1	0	0	8 (32%)
Breadth plus Depth and Moderately Coupled (Less Agile)	0	1	1	1	0	0	0	0	3 (12%)
Breadth plus Depth and Moderately Coupled (More Agile)	1	1	0	1	1	0	0	0	4 (16%)
Total	3 (12%)	4 (16%)	3 (12%)	6 (24%)	4 (16%)	4 (16%)	0 (0%)	1 (4%)	25

Table 5. Cross-tabulation of Process Variants and Final Team Grade

Team	Grade	Process Variant	Mean Man Hours	Team	Grade	Process Variant	Mean Man Hours
1	C+	2	242.83 (δ = 15.012)	14	C+	1	268.33 (δ = 11.377)
2	B	3	298.71 (δ = 17.168)	15	B	2	311.50 (δ = 6.526)
3	B-	3	290.17 (δ = 27.142)	16	B-	3	277.50 (δ = 12.013)
4	A+/A	5	331.67 (δ = 20.194)	17	C+	3	280.00 (δ = 5.916)
5	D+	1	274.00 (δ = 20.940)	18	B	3	281.67 (δ = 18.859)
6	C+	1	294.17 (δ = 19.469)	19	B	5	337.50 (δ = 10.062)
7	A-	2	293.33 (δ = 11.450)	20	B-	5	261.83 (δ = 9.765)
8	A-	3	315.00 (δ = 8.931)	21	B+	1	290.00 (δ = 10.408)
9	A-	4	283.33 (δ = 21.082)	22	B+	2	313.33 (δ = 15.202)
10	B+	4	302.17 (δ = 16.817)	23	A+/A	3	301.50 (δ = 13.137)
11	B	4	254.83 (δ = 7.346)	24	A+/A	3	311.00 (δ = 9.798)
12	B	2	278.33 (δ = 10.775)	25	A-	5	305.00 (δ = 16.833)
13	B-	2	307.67 (δ = 12.534)	**Process Variant**: 1 = Traditional Waterfall Model, 2 = Breadth and Loosely Coupled, 3 = Depth and Tightly Coupled, 4 = Breadth plus Depth and Moderately Coupled (Less Agile), 5 = Breadth plus Depth and Moderately Coupled (More Agile)			

Table 6. Mean Man Hours Expended for Development Activities

This aided them in their subsequent development activities, including the facilitation of short programming iterations, concurrent methods and optional method fragments. It appears that the appropriate use of process variant designing built upon the three decisive factors identified earlier could have a positive impact on students' performance in the ESD course. Coupled with the fact that the students had found process variant designing to be useful, we may draw a preliminary suggestion that process variant designing could be beneficial to students.

7. CONCLUDING COMMENTS

We believe that, owing to the explicit injection of IS development process variant designing framework into ESD, the majority of the students were generally able to assemble an integrated process variant appropriate for tackling a multi-tier enterprise system development project. In particular, preliminary observation suggests that a process variant tailored to the specific needs of a project could be effective in improving students' performance in a course compared to the use of the traditional waterfall model (caution should be exercised here since part of the quantitative statistical analysis results was not significant). In other words, process variant designing can benefit students invaluably. This is a noteworthy achievement given that the project was set in an unfamiliar business domain context, and the students were under the pressure of severe time constraints as well as the demand for a high standard of skills requirements. On a separate note, we initially conceived the injection of process variant designing into the course curriculum with the broader pedagogical objective of equipping the students with the necessary prerequisite knowledge to eventually learn and apply SME. We firmly believe that this objective has been achieved and that students would be better equipped to tackle SME when they enter the industry after graduation.

We hope that our experience would be useful for other educators for application in their own courses. The gist of our framework may be replicated in other IS development courses of a similar nature to our ESD course. For instance, if structured system analysis and design is being used with a non-object oriented development platform, then our framework may be modified by replacing our method base in Table 1 with the appropriate method fragments and removing those methods in Table 2 that are not suitable for the structured approach. The Rational Unified Process that is intended to be used with the object oriented UML diagrams may be dropped, as an example. Even when the object oriented system analysis and design is being used, educators

can still customize the method base according to their respective needs.

One of the key limitations of the study is the small sample size due to the fact that the statistical analysis was performed at the team level and for only one semester of students. This intrinsic limitation is difficult to address because IS development, and consequently process variant designing, is fundamentally a collaborative effort involving all team members. Future research can attempt to address this limitation if the course enrollment size proves sufficiently large. Another limitation is that the final letter grades awarded to the teams might not be solely attributable to the choice of process variant. Although we had shown that development effort did not affect the final letter grade, there could be other underlying factors not examined in our present study. Future research can also focus on how our process variant designing framework may be refined to provide an even more realistic training to prepare students for SME. Finally, it would be useful to develop appropriate matrices to explicitly assess the viability of process variant designing in improving the IS development performance of students.

8. ACKNOWLEDGEMENTS

The authors would like to thank all members of the teaching team who gave invaluable assistance throughout the entire semester. The authors would also like to express their appreciation to the anonymous reviewers and editors for their insightful comments on an earlier version of this paper.

9. REFERENCES

Alexander, L. C. and Davis, A. M. (1991) "Criteria for Selecting Software Process Models," in Proceedings of the 15th International IEEE COMPSAC, pp. 521-528.

Ambler, S. W. (1998) Process Patterns: Building Large-Scale Systems Using Object Technology, Cambridge University Press.

Ambler, S. W. (1999) More Process Patterns: Delivering Large-Scale Systems Using Object Technology, Cambridge University Press.

Aydin, M. N., Harmsen, F., Van Slooten, K and Stegwee, R. A. (2005) "On the Adaptation of an Agile Information Systems Development Method," Journal of Database Management, Vol. 16, No. 4, pp. 24-40.

Beck, K. (2000) Extreme Programming Explained, Addison-Wesley.

Becker, J., Janiesch, C. and Pfeiffer, D. (2007) "Reuse Mechanisms in Situational Method Engineering," in Proceedings of the IFIP WG8.1 Working Conference, Situational Method Engineering Fundamentals and Experiences (ME'07), Ralyté, J., Brinkkemper, S. and Henderson-Sellers, B. (eds.), IFIP International Federation for Information Processing, Vol. 244, Springer, Boston, pp. 79-93.

Bergner, K., Rausch, A., Sihling, M. and Vilbig, A. (1998) "A Componentware Development Methodology based on Process Patterns," in Proceedings of the 5th Annual Conference on the Pattern Languages of Programs.

Booch, G. (1991) Object-Oriented Design with Applications, Benjamin/Cummings.

Brinkkemper, S. (1996) "Method Engineering: Engineering of Information Systems Development Methods and Tools," Information and Software Technology, Vol. 38, No. 4, pp. 275-280.

Brinkkemper, S., Saeki, M. and Harmsen, F. (1998) "Assembly Techniques for Method Engineering," in Proceedings of the 10th International Conference on Advanced Information Systems Engineering (CAiSE 1998), Lecture Notes in Computer Science 1413, Thanos, C. (ed.), Springer-Verlag, Berlin, Heidelberg, pp. 381-400.

Cameron, J. (2002) "Configurable Development Processes," Communications of the ACM, Vol. 45, No. 3, pp. 72-77.

Carroll, J. (2003) "The Process of ISD Methodology Selection and Use: A Case Study," in Proceedings of the 11th European Conference on Information Systems (ECIS 2003), Naples, Italy, Paper 50.

Carroll, J., Rowlands, B., Standing, C., Frampton, K. and Smith, R. (2006) "A Framework for Redesigning ISDMs to Enhance Global Information Infrastructure," in Proceedings of the 2006 European and Mediterranean Conference on Information Systems (EMCIS 2006), Costa Blanca, Alicante, Spain, Paper C31.

Charette, R. N. (1989) Software Engineering Risk Analysis and Management, McGraw Hill, New York.

Davis, A., Bersoff, E. and Comer, E. (1988) "A Strategy for Comparing Alternative Software Development Life Cycle Models," IEEE Transactions on Software Engineering, Vol. 14, No. 10, pp. 1453-1460.

Firesmith, D. and Henderson-Sellers, B. (2002) The OPEN Process Framework: An Introduction, Addison-Wesley.

Fitzgerald, B, Hartnett, G. and Conboy, K. (2006) "Customising Agile Methods to Software Practices at Intel Shannon," European Journal of Information Systems, Vol. 15, pp. 200-213.

Fitzgerald, B., Russo, N. L. and O'Kane, T. (2003) "Software Development Method Tailoring at Motorola," Communications of the ACM, Vol. 46, No. 4, pp. 65-70.

Gnatz, M., Marschall, F., Popp, G., Rausch, A. and Schwerin, W. (2001) "Towards a Software Development Process Based on Process Patterns," in Proceedings of the 8th European Workshop on Software Process Technology (EWSPT 2001), Lecture Notes in Computer Science 2077, Ambriola, V. (ed.), Springer-Verlag, Berlin, Heidelberg, pp. 182-202.

Harmsen, F. (1997) "Situational Method Engineering," Unpublished Dissertation, Moret Ernst & Young Management Consultants, the Netherlands.

Harmsen, F., Brinkkemper, S. and Oei, J. L. H. (1994) "Situational Method Engineering for Information System Project Approaches," in Proceedings of the IFIP WG8.1 Working Conference on Methods and Associated Tools for the Information Systems Life Cycle, pp. 169-194.

Henderson-Sellers, B. (2003) "Method Engineering for OO Systems Development," Communications of the ACM, Vol. 46, No. 10, pp. 73-78.

Henderson-Sellers, B. and Serour, M. K. (2005) "Creating a Dual-Agility Method: The Value of Method Engineering," Journal of Database Management, Vol. 16, No. 4, pp. 1-23.

Iivari, J., Hirschheim, R. and Klein, H. K. (2000-2001) "A Dynamic Framework for Classifying Information Systems Development Methodologies and Approaches," Journal of Management Information Systems, Vol. 17, No. 3, pp. 179-218.

Iivari, J. and Koskela, E. (1987) "The PICO Model for

Information Systems Design," MIS Quarterly, Vol. 11, No. 3, pp. 401-419.
Jeyaraj, A. and Sauter, V. L. (2005) "Information System Development Methodologies as Learning Systems," in Proceedings of the 11th Americas Conference on Information Systems (AMCIS 2005), pp. 3091-3095.
Kim, Y., Hsu, J. and Stern, M. (2006) "An Update on the IS/IT Skills Gap," Journal of Information Systems Education, Vol. 17, No. 4, pp. 395-402.
Klooster, M., Brinkkemper, S., Harmsen, F. and Wijers, G. (1997) "Intranet Facilitated Knowledge Management: A Theory and Tool for Defining Situational Methods," in Proceedings of the 9th International Conference on Advanced Information Systems Engineering (CAiSE 1997), Lecture Notes in Computer Science 1250, Springer-Verlag, Berlin, Heidelberg, pp. 303-317.
Lemmen, K., Mulder, F. and Brinkkemper, S. (1999) "An Empirical Study on the Educational Effects of a Course in Method Engineering for Information Systems," Education and Information Technologies, Vol. 4, No. 2, pp. 181-202.
Mirbel, I. and Ralyté, J. (2006) "Situational Method Engineering: Combining Assembly-based and Roadmap-driven Approaches," Requirements Engineering, Vol. 11, No. 1, pp. 58-78.
Necco, C. R., Gordon, C. L., and Tsai, N. W. (1987) "Systems Analysis and Design: Current Practices," MIS Quarterly, Vol. 11, No. 4, pp. 461-470.
Neuman, W. L. (2006) Social Research Methods: Qualitative and Quantitative Approaches, 6th Edition, Allyn and Bacon, Boston, MA, pp. 246-271.
Nidumolu, S. R. and Knotts, G. W. (1998) "The Effects of Customizability and Reusability on Perceived Process and Competitive Performance of Software Firms," MIS Quarterly, Vol. 22, No. 2, pp. 105-137.
Noll, J. (2003) "Flexible Process Enactment Using Low-Fidelity Models," in Proceedings of the 7th IASTED International Conference on Software Engineering and Applications, ACTA Press, Anaheim, CA, pp. 675-680.
Ralyté, J. (2002) "Requirements Definition for the Situational Method Engineering," in Proceedings of the IFIP WG8.1 Working Conference on Engineering Information Systems in the Internet Context, pp. 127-152.
Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorensen, W. (1991) Object Oriented Modeling and Design, Prentice Hall, Englewood Cliffs, NJ.
Schneider, J. G. and Johnston, L. (2003) "eXtreme Programming at Universities – An Educational Perspective," in Proceedings of the 25th International Conference on Software Engineering (ICSE 2003), pp. 594-599.
Schuh, P. (2005) Integrating Agile Development in the Real World, Charles River Media.
Serour, M. K. and Henderson-Sellers, B. (2004) "Introducing Agility: A Case Study of Situational method Engineering Using the OPEN Process Framework," in Proceedings of the 24th Annual International Computer Software and Applications Conference (COMPSAC 2004), IEEE Computer Society Press, Los Alamitos, CA, pp.50-57.
Slooten, K. van and Brinkkemper, S. (1993) "A Method Engineering Approach to Information Systems Development," in Proceedings of the IFIP WG8.1 Conference on Information Systems Development Process, North-Holland, Amsterdam.
Song, X. and Osterweil, L. J. (1998) "Engineering Software Design Processes to Guide Process Execution," IEEE Transactions on Software Engineering, Vol. 24, No. 9, pp. 759-775.
Sutton, S. M., Heimbigner, D. and Osterweil, L. J. (1990) "Language Constructs for Managing Change in Process-Centered Environments," in Proceedings of the 4th ACM SIGSOFT Symposium on Software Development Environments, Irvine, CA, pp. 206-216.
Tolvanen, J.-P. (1998) "Incremental Method Engineering with Modeling Tools: Theoretical Principles and Empirical Evidence," Unpublished Doctoral Dissertation, University of Jyväskylä, Retrieved January 15, 2009, from http://users.jyu.fi/~jpt.
Turner, C. R., Fuggetta, A., Lavazza, L. and Wolf, A. L. (1999) "A Conceptual Basis for Feature Engineering," Journal of Systems and Software, Vol. 49, pp. 3-15.
Van Gurp, J., Bosch, J. and Svahnberg, M. (2001) "On the Notion of Variability in Software Product Lines," in Proceedings of the Working IEEE/IFIP Conference on Software Architecture (WICSA 2001), pp. 45-54.
Van Vliet, P. J. A. and Pietron, L. R. (2006) "Information Systems Development Education in the Real World – A Project Methodology and Assessment", Journal of Information Systems Education, Vol. 17, No. 3, pp. 285-293.
Wynekoop, J. L. and Russo, N. L. (1995) "System Development Methodologies: Unanswered Questions and the Research-Practice Gap," Journal of Information Technology, Vol. 10, No. 2, pp. 181-190.

AUTHOR BIOGRAPHIES

Wee-Kek Tan is an instructor and Ph.D. candidate at National University of Singapore. He holds a B.Comp. in Information Systems (First Class Honors) from the National University of Singapore. His research interests include social computing, the design of online decision aid, human-computer interaction, and information systems development and education. His research has been published in conferences such as ACM SIGMIS Computer Personnel Research Conference, IFIP Working Group 8.2 Working Conference, European Conference on Information Systems, and Americas Conference on Information Systems.

Chuan-Hoo Tan is an assistant professor of Information Systems at City University of Hong Kong. He holds a Ph.D. in Information Systems from the National University of Singapore. His research interests include the design and evaluation of consumer-based decision support interfaces, electronic commerce, and technology adoption. His work has been published in journals such as Information Systems Research, IEEE Transactions on Engineering Management, Annals of Operations Research, Information & Management, Electronic Markets, and Communications of the ACM as well as conferences such as International Conference on Information Systems.

Information Systems & Computing
Academic Professionals

STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

ISSN 1055-3096