B.Comp. Dissertation

Suitability of Plan-Based vs. Agile Methodologies in ISD Projects: The Power of Modularity

By
Yeo Yi Nah

Department of Information Systems

School of Computing

National University of Singapore

2013/2014

B.Comp. Dissertation

Suitability of Plan-Based vs. Agile Methodologies in ISD Projects: The Power of Modularity

By
Yeo Yi Nah

Department of Information Systems

School of Computing

National University of Singapore

2013/2014

Matriculation Number: A0075002X

Project No: H175030

Advisor: Associate Professor Hahn Jungpil

Deliverables:

Report: 1 Volume

Abstract

Recent surveys have shown that ISD project success rates are particularly low, and studies have shown that to ensure the smooth delivery of ISD projects, formal methodologies have to be in place as a form of structured development process. In recent years, organizations have been using a wide array of ISDMs – both plan-based and agile, such as waterfall, scrum and XP. However, as ISD projects often have different characteristics such as size, scope and complexity, numerous research have been done in hope to help organizations better choose an ISDM for projects. Nevertheless, research has not taken into consideration the problem modularity of ISD projects, which is an important factor that contributes largely to project architecture. In this research paper, we utilize the *NK* fitness landscapes model to examine the effects of problem modularity alongside various project environmental factors, and aim to answer the question: under various environmental factors, which ISDM should an ISD team adopt under various degree of problem modularity?

Keywords: modularity, software development methodologies, waterfall model, agile software development

List of Figures

Table of Contents

1. Introduction

Modern organizations are increasingly depending on IS largely for enhancing efficiency and competitive advantages. This phenomenon is coupled with the prevalence of information systems development (ISD) efforts within organizations (Xia & Lee, 2005). A fundamental goal of ISD projects is the delivery of high-quality systems that fits the needs and requirements of stakeholders. Studies have shown that one of the major challenges in ISD is the determination and delivery of system requirements, and this has led to the difficulty in reaching project success (Phariss, 2006). For example, a report of the Standish Group International (2009) on projects success rates shows that as much as 24% of all projects were complete failures (cancelled prior to completion or delivered but never used), 44% were challenged (late, over budget, and/or with less than the required features and functions) and only 32% were considered successful (delivered on time, on budget, with required features and functions). Since the 1980s, studies have shown that to ensure the smooth delivery of systems projects while containing risks and challenges, formal methodologies have to be in place as a form of structured development process (Brooks, 1975). Hence, it is important to choose an appropriate ISDM that increases a project's success rate, as the use of an adequate methodology plays an important role in developing software, to assure that it is delivered within schedule, within cost, and meets stakeholder requirements and needs (Geambasu et al, 2011). Furthermore, choosing an inadequate ISDM, or an ISDM that does not suit the characteristics of a project (in terms of processes, resources allocated, complexity, scope and size) can hinder the success of the development project. However, the existing literature has only broadly investigated the factors (e.g., project size, complexity, and organizational structure) that affect the selection of ISDM, most of which are largely project managerial in nature. Referring to the field of product engineering and development, while it is important to look at the managerial aspects to choose an appropriate development framework, a product's architecture and design is also central to the selection of an appropriate development methodology to maximize development configuration effectiveness (Yan, et al., 2007). Thus considering the area of ISD projects, it will be beneficial to look at the more structural aspect of a project – the degree of problem modularity. Studies into modularity have shown that by implementing formal methodologies that suits the modular nature of a project, further benefits can be made in terms of project quality and

performance, and will be discussed further (Yan, et al., 2007). Thus, examining project problem modularity can potentially provide us with insights to which ISDM might be more suitable for projects of various modularity levels and why the superiority.

Traditionally, the sequential waterfall methodology has been widely adopted. However, in recent years, the industry has seen an evolution towards the agile – or iterative – methodologies, comprising of models such as Xtreme Programming, Crystal and Scrum. It is noted that the traditional sequential waterfall methodology can be used with success largely for developing systems for which the requirements are clearly defined from the beginning of the project (Geambasu et al, 2011), whereas agile methodologies develop software in an incremental manner in numerous iterations while adapting to requirements changes. Proponents of the traditional waterfall methodologies have argued that planning and designing is more straightforward as developers and users agree on system requirements early in the project, and makes progress more easily measured. It is also argued that systems can be designed completely and more carefully, based upon a more complete understanding of all system deliverables, reducing the likelihood of the "piecemeal effect"[1]. Proponents have also criticized the iterative nature of agile development may lead to an overall reduction in system quality as there is less emphasis on understanding the system as a whole early in the project, and this becomes more pronounced in larger-scale implementations or systems that include high levels of integration. In contrast, proponents of the agile methodologies argue that agile developments are more user-focused, with highly quality software with better customer value due to the frequent requirements adaptations throughout the development process. Thus, proponents criticized the effectiveness of requirements and the possibility of delivering a dissatisfactory system due to the inflexibility to changes of the waterfall methodology. While there have been some discussions about which methodology is better for which projects under conditions such as project size and complexity, level of system integration, problem modularity, and the extent of requirements changes, there is a generally a lack of concrete and rigorous comparisons across methodologies. The arguments from proponents of both methodologies from practice present discussions and hypotheses without much principled arguments, thus it is tough to determine which

[1] The "piecemeal effect" is a development phenomenon that can occur as pieces of code are defined and subsequently added to an application where they may or may not fit well.

ISDM is superior under the various project conditions and the reasons behind the superiority.

Similarly, academic research has also focused broadly on selecting ISDM for projects based on numerous aspects of a project such as size, project complexity, development team and organizational structure (Vavpotič & Vasilecas, 2012). However, after 30 years of researching on how to better choose an appropriate ISDM for projects, the same question still surfaces within the research field. As ISD projects are complex in nature, choosing an adequate and appropriate ISDM that will deliver the best value amongst the wide array of options is a tough decision. Existing research have focused broadly on several project characteristics such as project size, scope and complexity, as well as organizational characteristics such as organizational structure. However, an important project characteristic that is untouched by existing research is problem modularity of ISD projects. As problem modularity contributes largely to project architecture, examining this different aspect of a project may provide us with interesting insights to the existing problem. Studies have shown that project modularity brings about vast advantages such as flexibility and rapid innovation, better management of complexity and the accommodation of future uncertainty. Modern organizations have since been embracing this "power of modularity" in ISD projects (Baldwin & Clark, 2004, and Ulrich & Eppinger, 1999). However, the modularity configuration differs for each project. Furthermore, proponents of modularity have also described the use of modularity as a problem-solving strategy by making complexity manageable by enabling development at the level of modules, rather than the entire system, and in parallel (Baldwin & Clark, 2001, and Brusoni et.al., 2007). Similarly, the ISD process represents a problem-solving process of searching for configurations that add value to the project performance (Davis, Bringham, & Eisenhardt, 2007). Since both concepts are fundamentally conceptualized as problem-solving strategies, examining the problem modularity feature of ISD projects further may yield potential insights to choosing an appropriate ISDM for ISD projects for the enhancement of project success rates.

The notion of modularity is central in the design and production of software artifacts, especially for large and complex projects[2]. Modularity is a general set of design principles that involves breaking up the system into discrete chunks that communicate with each other through standardized interfaces or rules (Langlois, 2002). Conceiving the design of a complex software artifact as a modular system means to apply the basic principle of "information hiding" that prescribes treating software modules as opaque entities. In essence, modularity aids in managing uncertainty and complexity (Kässi, Leisti, & Puheloinen, 2008). In recent years, the widespread adoption of object oriented languages and the diffusion of component based development as well other popular trends in software engineering have affirmed at large this information hiding principle and the paradigm of modularity as common software and system development practices, aimed at speeding up the development process, increasing innovation and increasing the success rates and quality of ISD projects (Narduzzo & Rossi, 2003).

Theoretically, software modularity affects software development and software quality, as the modular approach facilitates task decomposition, which is a strategy for efficiently organizing the work necessary to create deliverables. Thus, in theory, the degree of software modularity is expected to be positively related to software quality and thus, ISD project performance (Simon, 1996, and Conley & Sproull, 2009). However, to reap the potential enhancement in ISD project performance brought about by modularity, it is essential to implement an appropriate formal development methodology that fits the modular nature of the ISD project. Studies into modularity have shown that by implementing formal methodologies that suits the modular nature of a project, further benefits can be made in terms of time, cost, quality and ultimately, performance. Thus, there is an explicit need within industry to develop and implement appropriate methodologies that are able to guide the direction of the whole design and development process using modular concepts in a systematic manner (Yan, et al., 2007, and Duffy, Smith, & Duffy, 1998). With reference to the electronics industry, a study has shown that a difference in product quality and cost of up to 64 times is present when a development methodology that suits the modular practices of the project is implemented, as compared to the reliance on designers' natural inclinations (Yan, et al.,

[2] Modularity has been receiving an increasing amount of attention in a variety of fields. In recent years, modular approaches have been widely adopted by software engineering projects, and has been extensively discussed in various literatures (Baldwin & Clark, 2000, Brooks, 1975, and Narduzzo & Rossi, 2003).

2007, and Duffy & Ferns, 1998). In hardware engineering, the creation of the end product is heavily process driven, and very often, the defined process leads to a highly successful product creation. In contrast, software engineering aims to extract functionalities out of hardware designs, and involves a much higher level of complexity due to flexibilities. Since modularity enhances the performance of hardware engineering, it is therefore intuitive that similarly, in the field of software development, modularity should also have a differential impact for different ISDM implemented in an ISD project.

To improve the success rates of ISD projects, it is thus beneficial to investigate the link between the two concepts, how different types of ISDM and various degrees of modularity can affect the effectiveness and success of an ISD project. This study aims to explore this linkage and its associated performance relationship. The main research focus of this paper is to investigate which form of ISDM (waterfall or agile) leads to better ISD project performance under various degrees of problem modularity. While it is important to select an adequate ISDM for the ISD project, is it also important to grasp the optimal degree and extent of modularity for the best project quality and performance, to aim for better ISD project success rates. This leads to the question of the linkage between the two concepts – which ISDM better fits each degree of modularity.

However, it is essential to note that ISD projects exist in different forms – projects have variations in terms of size, scope, complexity and resources allocated to them. These project elements will affect the system development process. This paper will take into consideration the impact of these project elements while examining the dual linkage between ISDM and modularity to enhance the robustness of the research.

The specific objectives of this research paper are the following:

- understand the impact of modularity on ISDM,
- identify the contextual factors that impact the effectiveness of different ISDM (project size, scope complexity and resources), and
- understand the impact of modularity on ISDM under the effects of contextual factors.

2. Research Approach & Theoretical Background

As we investigate ISD projects in general, an important aspect of this research is to ensure that results have high generality such that the theoretical insights can apply to more than just to one or few specific forms of organizations. Amongst the wide array of research tools such as surveys, case studies, mathematical and analytical models, we adopt the computational modeling and simulations approach to model the design problem solving in ISD projects. Field studies such as surveys and case studies enjoy the luxury of realism, with the data collected being richer and of greater depth. However, field studies may be limited in generalizability and in what can be observed and measured since it is difficult to manipulate all variables of interest in a field setting. Conversely, mathematical / analytical models allow formal analysis with rigor but must frequently rely on drastically simplified representations of organizations for analytical tractability and, as a result, cannot faithfully represent the richness of actual organizations. Nonetheless, the simulation methodology frees manipulation of the project elements of interest in a formal model that incorporates a greater number of interdependent elements than is possible with a closed-form analytical approach, which helps acquire theoretical insights through various combinations of experimental conditions for greater generalizability (Amaral & Uzzi, 2007, and Davis, Bringham, & Eisenhardt, 2007). In particular, this generalizability is crucial in this research to ensure the applicability to ISD projects within the industry.

There is a wide variety of simulation approaches such as system dynamics, NK fitness landscapes, genetic algorithms, cellular automata and stochastic processes. In this research, we employ the NK fitness landscapes simulation approach and extend it to model ISD processes while considering the characteristics of modularity inherent in projects (Kauffman, 1993, and Kauffman & Weinberger, 1989). The NK fitness landscapes model is most appropriate when project adaptation can fundamentally be conceptualized as problem solving or search (Davis, Bringham, & Eisenhardt, 2007). In this research, the ISD process can be adequately conceptualized as a problem solving process, which will be discussed later.

A key concept of the NK fitness landscape approach is the fitness landscape that is created by assigning performance values (fitness) to different project configuration

decisions. Also, the theoretical logic of this approach focuses on the adaptation of a system using search strategies to find an optimal point on a fitness landscape. The fitness landscape represents the external decision environment the agents must interact with, where we can create landscapes exhibiting varying degrees of environmental complexity or "ruggedness"[3] (Levinthal, 1997). The concept of "ruggedness" will be further elaborated later in the modeling of the project performance landscape section. The model allows the creation of fitness landscapes representing differing degrees of project modularity onto which computational agents (ISD team) set to follow behavioral rules for adaptation (e.g., waterfall vs. agile) can be seeded and their adaptation behaviors can be observed. By simulating many different agents' behaviors, we can analyze the statistical properties of the adaptive problem solving processes (waterfall and agile ISDM) under different environmental contexts (e.g., project modularity, project size, scope, complexity and resources).

In this research, we aim to understand the ISD process of projects with different modularity, thus we adopt a process-oriented view. ISD is conceptualized as a systemic work activity involving the process of system analysis, design and development (Korpela, Mursu, & Soriyan, 2002). This process can be conceptualized as a problem solving process where the agent – the ISD team – performs adaptive search strategies for an ISD project configuration that delivers value to the ISD project. Specifically, the ISD team identifies a performance gap between the existing and desired states of the project configuration, and carry out activities such as detailed user requirements gathering, generating alternative designs and solutions in the attempt to reduce the performance gap and achieve higher levels of ISD project performance. This process of search is adaptive and experiential. The ISD team is assumed not to have perfect knowledge of the true structure of the complexity of the problem space (for example, the interdependencies among decision variables), but the team is able to infer the value of different system designs when the specific configurations are considered (Cerveny, Garrity, & Sanders, 1990). Referring back to the NK model, this search process is modeled by the agent (ISD team) searching and adapting to different configurations on

[3] If the project fitness is highly dependent, that is, the value of a particular configuration of the project depends on a variety of other configurations of the project, then the fitness landscape will tend to be rugged with many peaks.

the landscape in search for more superior performance values, which will be elaborated in detail later in the modeling of ISD process section.

In addition, there are several main differences between the traditional waterfall methodology and agile methodology. The waterfall methodology can be conceptualized as one big project that involves a structured and sequential development process, whereas the agile methodology can be conceptualized as a set of numerous smaller projects that involve a flexible and collaborative develop process (Dyba & Dingsøyr, 2008). In this research, the differences between the two ISDM are modeled and simulated by the search process, which is divided into one or more modules when using different ISDM (waterfall and agile). In the most extreme case, when the waterfall methodology is adopted, search is performed on the whole problem space, which is regarded as one module, whereas when the agile methodology is adopted, search is performed on the subdivided problem space, which contains two or more modules.

3. Hypotheses

Theoretically, the architecture of an agile development methodology emphasizes on the structural resiliency of the system such that change and adaptation is possible throughout the development life cycle. Thus flexibility and change accommodation are essential in an agile model, and modularity becomes a critical to agile architecture, where the modular approach facilities the understanding of the impact of change and change management as well as implementation at the module level. Hence, we hypothesize:

H1: As the degree of problem structure modularity increases, the agile methodology will outperform the waterfall methodology.

Furthermore, as modularity points to the notion of breaking down large complex problems into smaller units, modularity thus aids in managing uncertainty and complexity (Kässi, Leisti, & Puheloinen, 2008) Also, as a problem-solving strategy, modularity increases efficiency in projects with larger size and scope as well as complexity, by facilitating task decomposition and allowing teams to work on smaller portions of the project at once, thereby enhancing productivity and reducing overhead (Oracle, 2007). As a continuation from H1, we hypothesize:

H2: With increasing project size, scope and complexity, projects with higher degrees of modularity that adopt the agile methodologies should lead to better ISD project performance relative to the performance of the waterfall methodology.

4. Model

The model setup requires the specification of four features: (1) the representation of a project's performance landscape; (2) the various modular design choices; (3) the process of the ISDM applications; and (4) the process of local search as project's process of adaptation on the landscape.

4.1. Modeling the Project's Performance Landscape

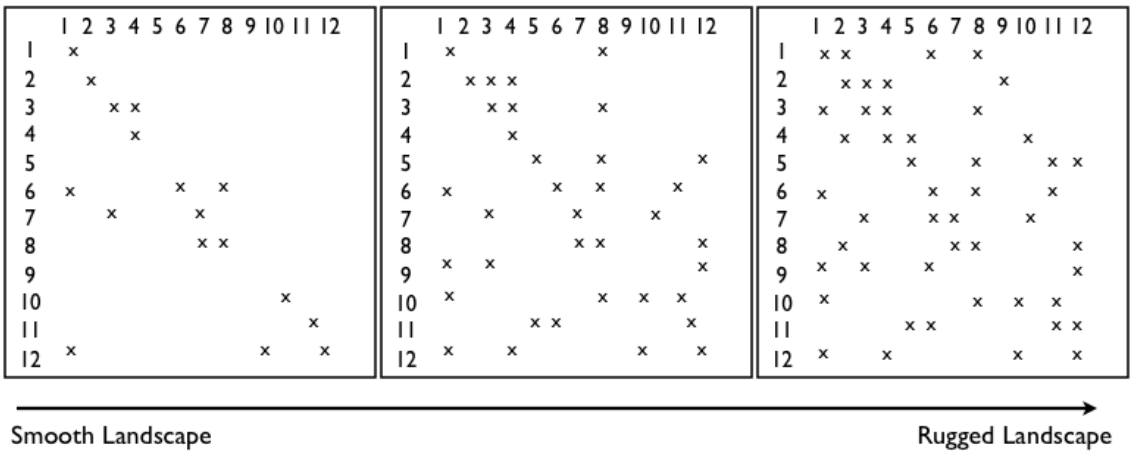
In this paper, the *NK* fitness landscapes model (Kauffman, 1993) is adopted to explore the performance outcome of projects given various behavioral rules and degrees of environmental uncertainty. This paper also adopts a project-focused perspective instead of a firm-focused perspective by modeling the performance landscape, using the *NK* model, specific to projects undertaken in a typical firm.

A project p, is represented by a set of N decision variables, $p = \{d_1, d_2, \ldots, d_N\}$, where each decision d_i can take on either one of the two possible values (0, 1). For example, in the case of the development of a sales and marketing system, d_1 can be the decision to enable customer order tracking ($d_1 = 1$) or not ($d_1 = 0$), d_2 can be the decision to enable the sending of push notifications ($d_2 = 1$) or not ($d_2 = 0$), and d_3 can be the decision to enable customer account access ($d_3 = 1$) or not ($d_3 = 0$). Each decision contributes to overall value of the project configuration and the value contribution of each decision depends not only on the choice made concerning that decision ($d_i = 0$ or 1) but also on choices regarding K other decisions. In other words, each decision variable may be tightly linked to other decision variables. Thus, the performance of the project depends on the performance contributions of all decision variables based on the interactions among them.

In the most extreme cases, where there are no linkages amongst the decision variables, and all decision variables are independent, each decision then contributes independently to the project performance – choice of altering one decision will not affect the performance contribution of other decisions. For example, the decision to enable customer order tracking or not does not depend on the decision of whether to enable the sending of push notifications. This results in a performance landscape that is smooth with a single peak. On the other hand, where there are strong linkages amongst decision

variables, then the project performance contributions of all decisions become interdependent – choice of altering one decision will affect the performance of other related decisions. This results in a performance landscape that is rugged and with multiple peaks. For example, the decision to enable customer order tracking will depend on the decision to enable the access of customer accounts by the customer himself. These are the extreme cases where the interdependencies among the decision variables within the project is equals to 0 and (N – 1), and the landscape is the smoothest and most rugged, respectively. However, all values between 0 and (N – 1) are possible, and they represent the varying degrees of landscape ruggedness or project complexity. Figure 1 below shows the influence matrixes of varied interdependencies.

Figure 1. Influence Matrixes



In general, the performance contribution (c_i) of each decision variable (d_i) is dependent on its own condition as well as the conditions of its K other dependent decision variables, where

$$c_i = c_i(d_i \mid K\ other\ d_j),$$
$$i \neq j$$

Without loss of generality, we assume that all decisions carry equal weights of contributing to the overall performance of the project, thus the average performance of all the contributions of all decision variables is a good measure in this case. The overall project performance outcome, P, is

$$P = \frac{1}{N}\sum_{i=1}^{N} c_i$$

In addition, a software development project may involve uncertainties such as requirements uncertainty, as well as insufficient knowledge about business and technical needs. Thus, to better reflect ISD projects in reality, we also take project uncertainty into consideration, where project uncertainty can range from 0% to 100% when the project landscape is modeled. We model uncertainty by referring uncertainty as the imperfect assessment of the implications of making design choices – the higher the uncertainty, the higher possibility of inaccurate prediction of performance implications of moving to another point on the landscape. However, uncertainty will decrease overtime throughout the course of the ISD project as more feedback is obtained from the ISD team. As the project nears completion, uncertainty will approach zero.

4.2. Modeling the Design Structures

To model modularity, we arrange the interdependencies of the decision variables into clusters. Each cluster in the landscape represents a module. To examine the effects of various ISDM with respect to project modularity, we consider three design structures of varying degrees of problem modularity – perfect modular, imperfect modular, and non-modular (Ethiraj et al, 2008).

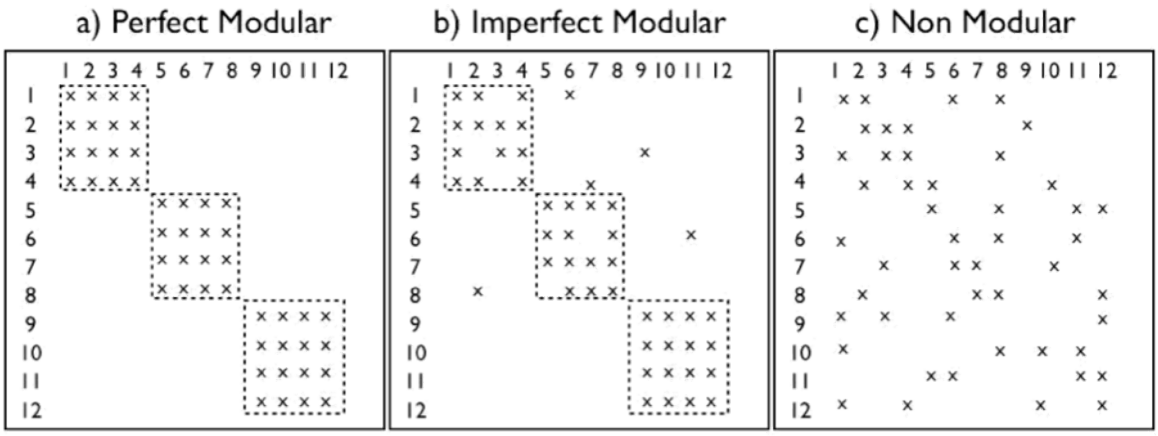
Each design structure comprises of the total number of decision variables N, the number of modules M, and the number of interdependencies R, i.e, $R = NK$.

All 3 design structures will have the same value of R (i.e., the same number of interdependencies spread across the various decision variables). However, the distributions of the interdependencies are different for each design structure, which

determines the degree of modularity, as shown in Figure 1. For each x in the cell d_{ij}, it represents the dependency between the decision variables d_i and d_j (i.e., the performance contribution of d_i depends on the value of d_j).

For the perfect modular and imperfect modular design structures, we will assume the same size of all modules, where each module will consist of decision variables. In the perfect modular design structure (see Figure 1a), referring to figure 1a, the R interdependencies are distributed across M modules such that each module consists of exactly $\frac{N}{M}$ tightly coupled decision variables. In a perfect modular structure, the R interdependencies form a block-diagonal structure. The imperfect modular design structure[4] (see Figure 1b) differs from that of a perfect modular structure such that for each of the M modules, a number of intra-module interdependencies X are randomly chosen and removed, and randomly reintroduced as inter-module interdependencies. Thus, the value of R remains constant while the distribution of R is changed. However, it is important to note that the number of intra-module interdependencies will always be larger than the number of inter-module interdependencies for the imperfect modular structure. Finally, the non-modular design structure (see Figure 2c) has randomly distributed R interdependencies, where each cell has an equally likely chance of having an assigned interdependency.

Figure 2. Varying Degrees of Project Modularity (N = 12, M = 3, R = 36)



[4] Some authors have defined imperfect modular structures to incorporate two properties—hierarchy and near-decomposability (Ethiraj, Levinthal, & Roy, 2008). However, this concept is not explored in this paper.

4.3. Modeling the ISD Process

In the experiments, the project teams engage in local experiential search – for an agile development process, local search represents the within-module incremental innovation attempts (the system contains several modules), whereas local search for a waterfall development process represents the incremental innovation attempts within the entire system (the system contains only one module). The agents (ISD teams) perform local search at the module-level, attempting to enhance the performance at module-level. However, such performance enhancements may not necessarily lead to system-level performance increase.

In each iteration, the agents (ISD team) select a neighboring decision variable at random (out of the decision variables considered for that iteration), and evaluate the performance decrease or increase by flipping the decision choice between 0 and 1. If the flipping of decision choice results in a performance increase, the improvement is adopted and implemented; else, it will be discarded.

4.4. Modeling the Application of ISDM

While agile methodologies consist of a wide range of variety such as Crystal and Scrum, in this paper, it is recognized that the key differences between the waterfall and agile methodologies are the use of iterative development styles as well as the scope focus – with the waterfall methodology regarding the ISD project as one big project, and the agile methodology regarding the ISD project as several small projects. Thus, in this paper, the application of waterfall and agile methodologies are examined broadly, focusing on the key contrasts between the two types of methodologies, without further classification into the various specific agile methodologies.

4.4.1. Agile Development Methodology

In this development methodology, change to only one attribute per module is permitted at once. The agile development methodology is applied through two main characteristics.

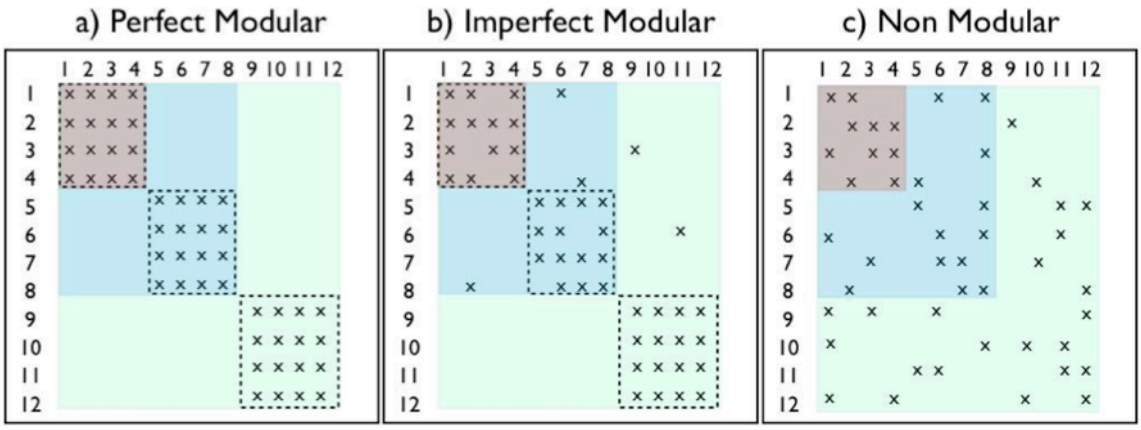
Firstly, there will be I iterations, where I is equal to M. in each iteration, a factor of the N decision variables which have not been considered for incremental improvement in previous iterations will be considered for incremental improvement. Upon performing

the search process amongst the factor of the N decision variables, the search process will take place again on the decision variables considered for the previous implementation. This is to model the flexibility to changes of modules within the agile development methodology. For example, for a design structure with 12 decision variables and 3 modules (i.e. $N = 12$ and $M = 3$), there will be 3 iterations. During each iteration, only one module, which has not yet been considered for incremental improvement, will be considered (i.e. only 4 new (previously not considered) sequential decision variables will be considered in each iteration).

Secondly, upon performing incremental improvement on the module, the agents are allowed to change one attribute of the module considered for the previous iteration(s) for performance improvement.

Referring to Figure 3, each iteration is represented by the red, blue and green boxes respectively. For example, the first iteration is represented by the red box, where the search process will only occur from d_1 to d_4. The second iteration is then represented by the blue box, where the search process will first occur from d_5 to d_8, then from d_1 to d_4 again. Similarly, the third iteration is represented by the green box, where the search process will first occur from d_9 to d_{12}, then from d_5 to d_8, and lastly from d_1 to d_4 again.

Figure 3. Agile Development Process



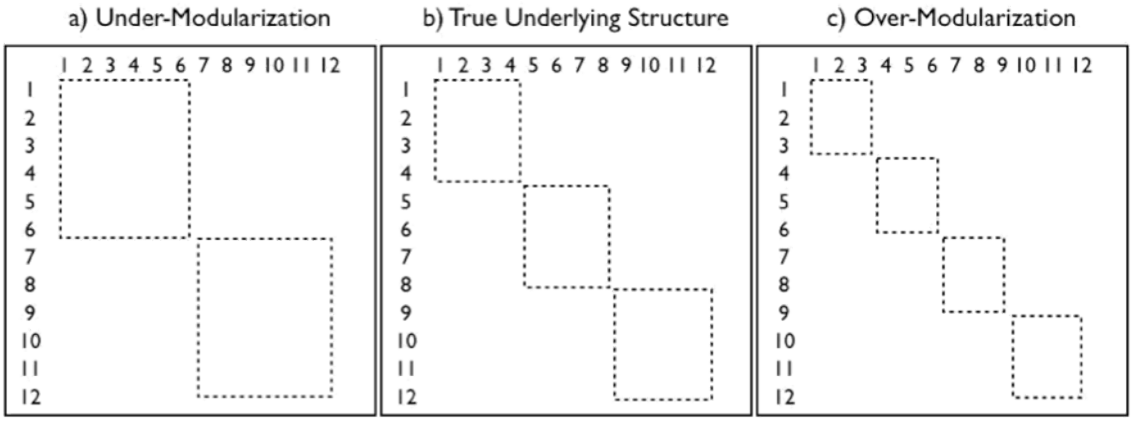
4.4.2. Waterfall Development Methodology

Similarly, the waterfall development methodology is applied such that only one change to one attribute can be made at once. However, in each period, all decision variables will be considered, in contrast with the agile methodology.

4.5. Modeling the True Underlying Design Structure

Considering the modular design structures (i.e., the perfect modular and imperfect modular structures), by varying the value of the number of modules M, we can examine the effects of over-modularity and under-modularity. This notion is based on the basis that agents (the ISD team) do not have perfect knowledge of the true design structure of the project (i.e. they do not have the knowledge of the optimal number of modules to have within a project). Thus, their guess can be classified as over-modularized, under-modularized, of perfectly modularized if the guess fits the true underlying design structure. Letting the true design structure be M^I, a design structure is considered to be over-modularized when $M > M^I$, and under-modularized when $M < M^I$. Referring to figure 3, figure 3b shows the true underlying structure ($M^I = 3$), Figure 4a shows the under-modularized structure ($M < M^I$) and figure 3c shows the over-modularized structure ($M > M^I$).

Figure 4. Representation of the True Underlying Structure



5. Experiments

In order to examine the various degrees of modularity, the two different kinds of development methodologies and their effect on ISD project performance, we perform three levels of experiments. The first level of experiment will represent the baseline, where we inspect the basic relationship between the types of methodologies and degrees of modularity. In the second level of experiments, we will manipulate one parameter in each case – number of modules, complexity, size and scope, as well as resource availability of the ISD project. Lastly, in the third level of experiments, we will manipulate more than one parameter in each case to model our experiments closer to reality.

To ensure the generalizability of the simulation results, all results are based on 50 independently generated fitness landscapes for each influence matrix (representing the design structure), with 20 ISD projects randomly seeded onto each fitness landscape. The average performances throughout 20 time periods are recorded[5]. The performance of the ISD project is measured as a portion of the highest performance attainable on each landscape[6] (Hahn & Lee, 2011).

5.1. Baseline Performance Relationship Between ISDM and Modularity

The first experiment examines the basic performance relationship solely between the software development methodologies and the degrees of modularity. In the first part of this experiment, the aim is to investigate the effectiveness of each type of methodology on the three degrees of modularity – perfect modular, imperfect modular and non-modular. We set $N = 16$, $K = 6$ and $M = 4$. For each degree of modularity, the distribution of the R interdependencies will be different, as explained in the model. We choose $N = 16$ to generate fitness landscapes corresponding to ISD projects of sufficient size. Also, this value is appropriate as we can vary the value of N to simulate

[5] See Appendix 1 for the adequacy of the experiential settings.

[6] In the *NK* landscapes simulations, the parameter *K* (representing the overall complexity of the landscape) determines the maximum fitness value within the landscape. Smooth landscapes (small *K*) generally have smaller maximum fitness value that are easier to attain, as compared to rugged landscapes (large *K*) which have larger maximum fitness values that are tougher to reach. Thus, when comparing performance across complexity levels (i.e., across *K*), it is appropriate to compare fitness values that are normalized to the maximum attainable performance levels rather than the raw fitness measures (Hahn & Lee, 2011).

projects of smaller and larger modules in the later experiments, while still maintaining the adequacy of a typical ISD project size.

The analyses of the basic performance relationship between the software development methodologies and the degrees of modularity are based on 14 experimental configurations of the ISD project – (1 x perfect modular design + 3 x random imperfect modular design + 3 x random non-modular design) x 2 ISDM applications = 14 experimental configurations.

Figures 5(a) to 5(c) represents a series of simulations that examine the average performances of the agile and waterfall ISDMs on projects with 3 degrees of modularity – perfect modular, imperfect modular and non-modular.

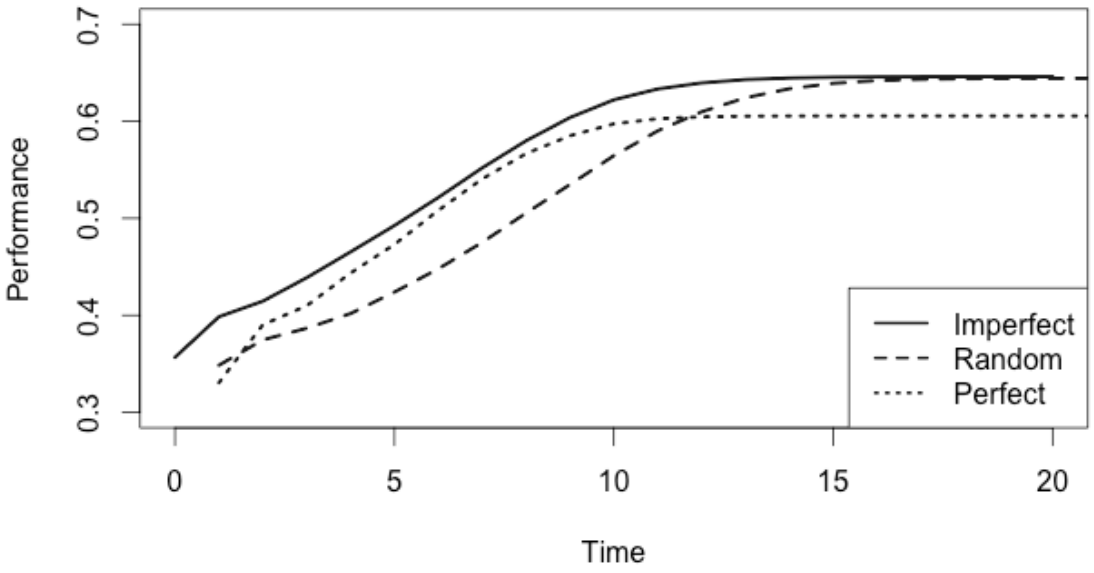
From Figures 5(a) and 5(b), we see that when the agile methodology is adopted, in terms of speed, perfectly modular is the first to plateau, followed by the imperfect and non-modular structures. In addition, the imperfect modular structure performed best in terms of peak performance. However, the peak performance for perfectly modular structure is significantly lower than both the imperfect modular and non-modular design structures when agile methodology is in use. Thus, although the perfectly modular structure has a faster initial performance improvement, the end performance attained is less satisfactory than the other designs. When the waterfall methodology is adopted, performance is relatively consistent for both imperfect modular and non-modular structures. In contrast, the peak performance for perfectly modular structure is relatively lower than the other two structures. The performances of all three design structures take similar time to plateau. From Figure 5(c), we see that the agile methodology performs better than waterfall methodology in terms of peak performance attainable for both the imperfect and non-modular structures. However, peak performance attainable for perfectly modular structure is consistent for both the agile and waterfall methodologies.

When a project is modularized, the adaptive and iterative nature of the agile methodology aids in producing projects of higher quality performance, which outperforms that of waterfall. However, when interdependencies between modules are absent, a modularized project structure seems to impede the performance of the project,

regardless of the ISDM used. As an adaptive development approach, agile operates in modules, allowing for change in requirements and adaptations of decisions. Perhaps due to the adaptive nature of agile, its benefits are not largely realized in the perfectly modular structure, as the landscape is smooth and there are zero interdependencies between modules. Thus, a certain extent of interdependencies between modules in a project has to be present for agile to realize its benefits.

Furthermore, relevant research has also demonstrated that as the number of local peaks proliferate in the performance landscape, it becomes harder for a project to attain the high peaks in the landscape and achieve a high performance. In particular, the research has shown that in the setting of a perfect modular design structure with $K = 3$, the average number of local peaks is significantly more than that of the imperfect-modular and non-modular design structures[7] (Rivkin & Siggelkow, 2007). Hence, due to a higher number of local peaks present in the landscape, the landscape of the perfect modular design is more rugged than the other two designs despite having the same K value. Thus, it is intuitive that it is harder for the perfect modular design to attain a high performance as compared to the other two designs.

Figure 5(a). Agile ISDM & Degrees of Modularity (N = 16, M = 4, K = 3)



[7] In the relevant research done by Rivkin and Siggelkow, the perfect modular, imperfect-modular and non-modular structures are equivalent to the block-diagonal structure, small-world and random structures respectively. The block-diagonal, small-world and random structures yield an average of 37, 24 and 23 local peaks respectively.

Figure 5(b). Waterfall ISDM & Degrees of Modularity (N = 16, M = 4, K = 3)

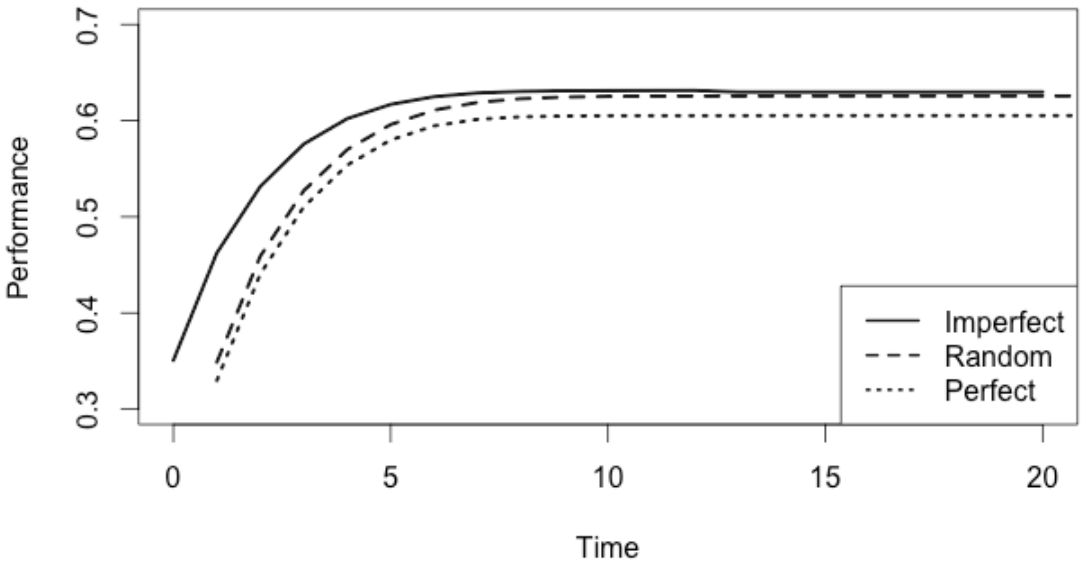
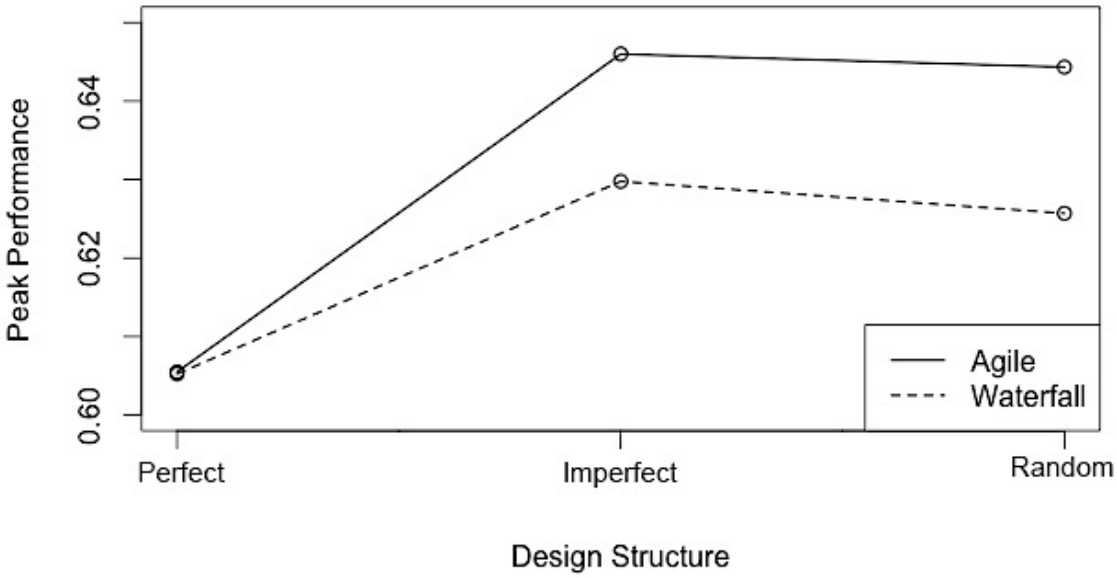


Figure 5(c). ISDMs & Degrees of Modularity (N = 16, M = 4, K = 3)



5.2. Performance Relationship Between ISDM and Modularity Under Conditions (Single Parameter)

In this section, we investigate the effects of the following conditions on the performance relationship between ISDM and various degrees of modularity:

- Under-modularization and over-modularization

- Low to high project complexity
- Small to large project scope and size
- Limited and unlimited resource availability

5.2.1. The Effects of Under and Over-modularization

This experiment will be performed on both the imperfect modular and non-modular design structures. We assume that the true underlying structure be $M = 4$, we keep $N = 16$, $K = 6$ constant, and vary the number of modules M', thereby varying the number of iterations I as well. We set M' as values 2 (for under-modularization), and 8 (for over-modularization).

The analyses of the performance relationship between the software development methodologies and under-modularization as well as over-modularization are based on 42 experimental configurations of the ISD project – (3 x random imperfect modular design + 3 x random non-modular design) x 3 levels of modularization x 2 ISDM applications = 36 experimental configurations.

Letting the true underlying structure be $M = 4$, Figures 5(a) to 5(d) represents a series of simulations that examine the average performances of the ISD projects using agile and waterfall on the imperfect modular and non-modular designs[8], when there is under and over-modularization of the project structure.

From Figures 6(a) and 6(b), we see that when the project structure is imperfect modular and the agile methodology is used, performance for the true underlying structure (i.e. $M = 4$) has the highest peak as compared to the under and over modularized structures. In particular, the under-modularized project yields a lower peak performance, and the over-modularized project takes a significantly longer time to peak. It can be deduced that when an ISD project becomes over-modularized, there are too few decision variables to be considered in each iteration. Thus, performance improvement slows down, and the benefits of adaptability are diminished. In contrast, when the waterfall methodology is used, performance is relatively consistent for all design structures. This demonstrates further that the effects of under and over modularization is negligible in

[8] The perfect modular structure is inapplicable in this case as a perfect modular structure is inexistent for M=8.

the case of the waterfall methodology, as all decision variables are considered at once, i.e, $M = 1$.

Figure 6(a). Agile ISDM & Imperfect Modular Structure (N = 16, M = 2, 4, 8, K = 3)

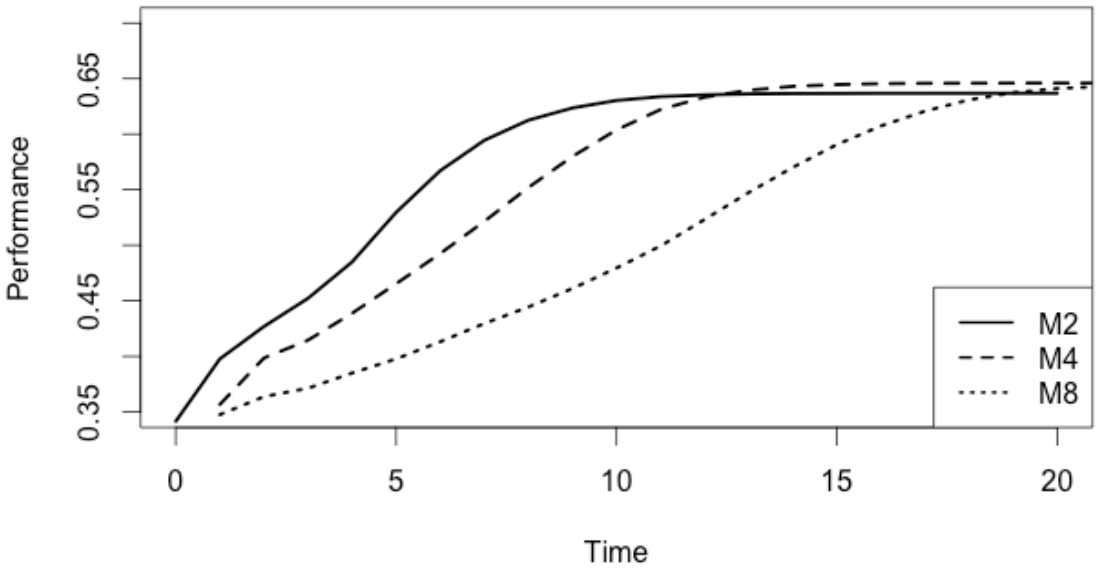
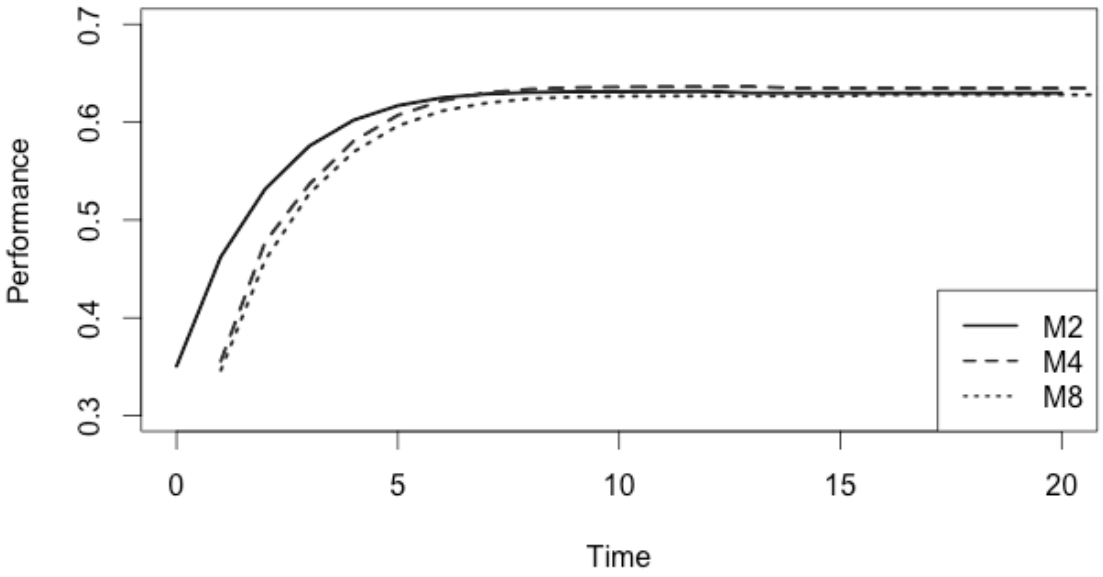


Figure 6(b). Waterfall ISDM & Imperfect Modular Structure (N = 16, M = 2, 4, 8, K = 3)



From Figures 6(c) and 6(d), we see that when the project structure is non-modular and the agile methodology is used, performance peaks are consistent for $M = 2$, 4 and 8. However, performance plateaus fastest when project structure is under-modularized and slowest when over-modularized. Similarly, for an over-modularized structure, there are

too few decision variables to be considered in each iteration, thus performance takes a significantly longer time to improve and peak. Similarly, when the waterfall methodology is used, performance is relatively consistent for all design structures.

Figure 6(c). Agile ISDM & Non-modular Structure (N = 16, M = 2, 4, 8, K = 3)

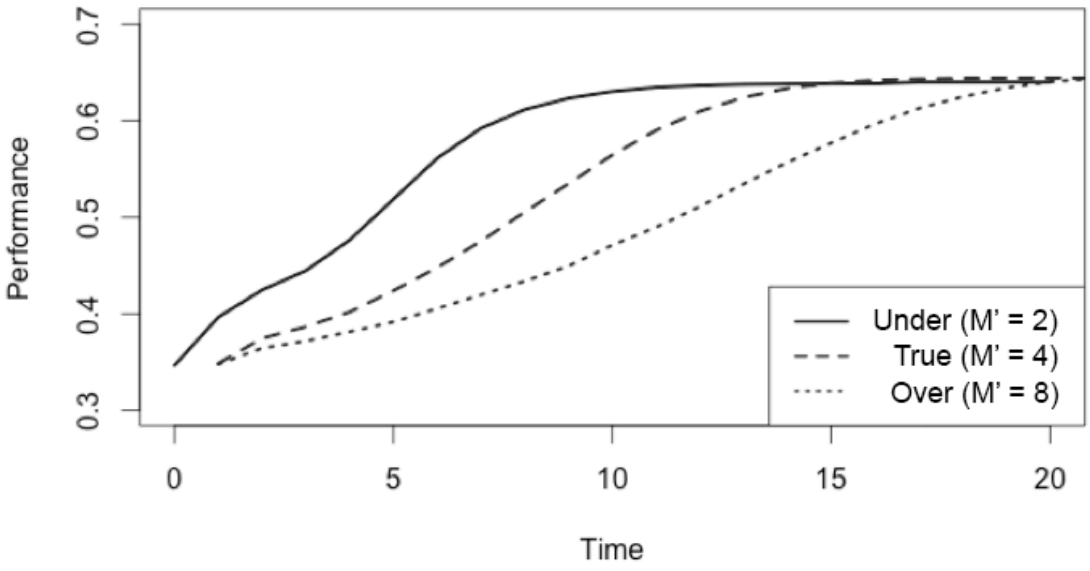
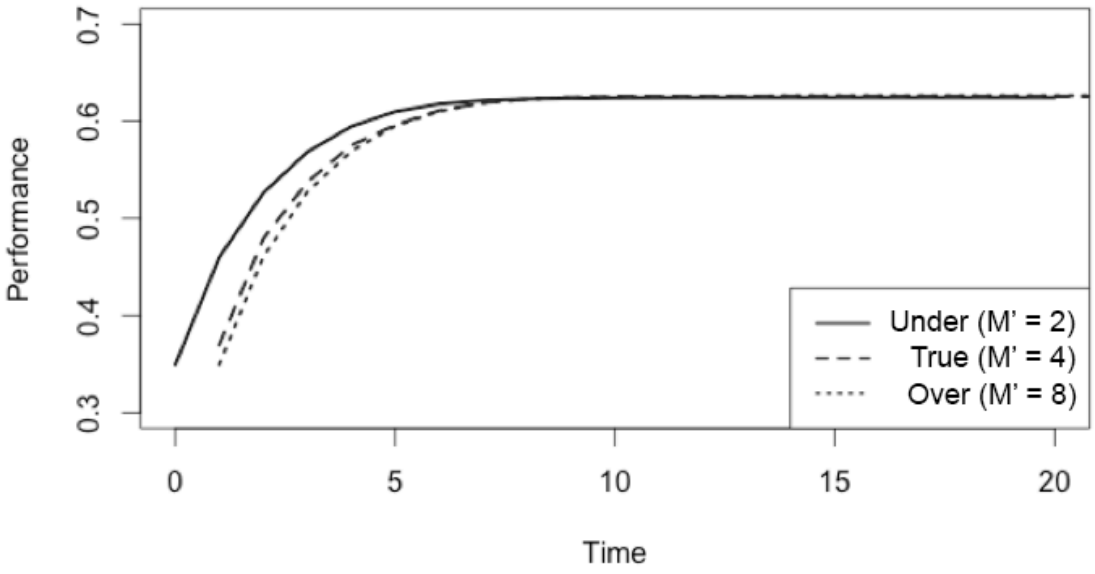


Figure 6(d). Waterfall ISDM & Non-modular Structure (N = 16, M = 2, 4, 8, K = 3)



5.2.2. The Effects of Varied Project Complexity

This experiment takes into account project complexity factor while examining the performance relationship between the methodologies and degrees of modularity. While we vary the design complexity by altering the distribution of the interdependencies, we vary the cognitive complexity by altering the environmental complexity of the landscape within the *NK* model. Hence, in this experiment, we set $N = 16$ and $M = 4$ constant. While varying the cognitive complexity (i.e. varying K in the *NK* model, where each decision variable can only have dependencies on $K - 1$ other decision variables). Thus, we set K as 3, 6, 8 and 10, with R holding the values 48, 96, 128 and 160 respectively.

The analyses of the performance relationship between the methodologies and degrees of modularity, taking into account project complexity, are based on 50 experimental configurations of the ISD project – [(3 x imperfect modular x 4 levels of K) + (3 x non-modular x 4 levels of K)] x 2 ISDM applications = 48 experimental configurations.

Figures 7(a) to 7(c) represents a series of simulations that examine the average performances of the ISD projects using agile and waterfall on the imperfect modular and non-modular designs, where the project complexities are varied from low to high.

From the results of the experiment, as complexity increases, the time taken by the projects to plateau (in terms of performance) decreases consistently for all cases as complexity increases. Considering the peak performances of the projects from Figures 7(b) and 7(c), we see that as complexity increases, peak performances decrease for both the imperfect modular and non-modular design structures, in both cases where the agile and waterfall methodologies are adopted respectively. Further, when agile is used, projects with an imperfect-modular design structure outperform that of a non-modular structure. In contrast, when waterfall is used, the performance gaps between imperfect-modular and non-modular close in as complexity reaches higher values (i.e. $K = 8$ and 10).

Figure 7(a). Time Taken for Performance to Plateau

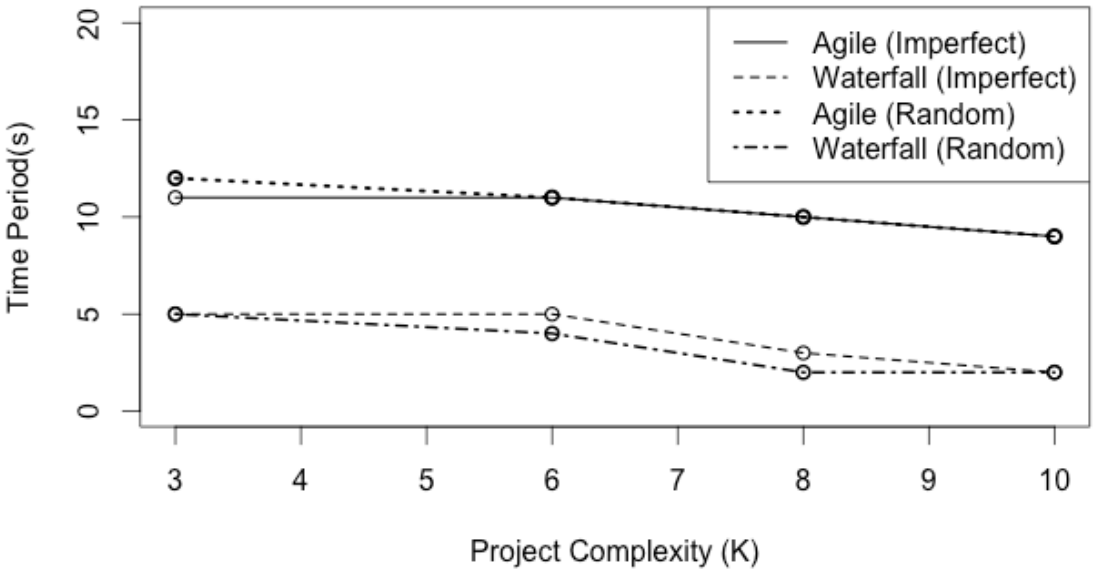


Figure 7(b). Agile ISDM with Varied Complexity (N = 16, M = 4, K = 3, 6, 8, 10)

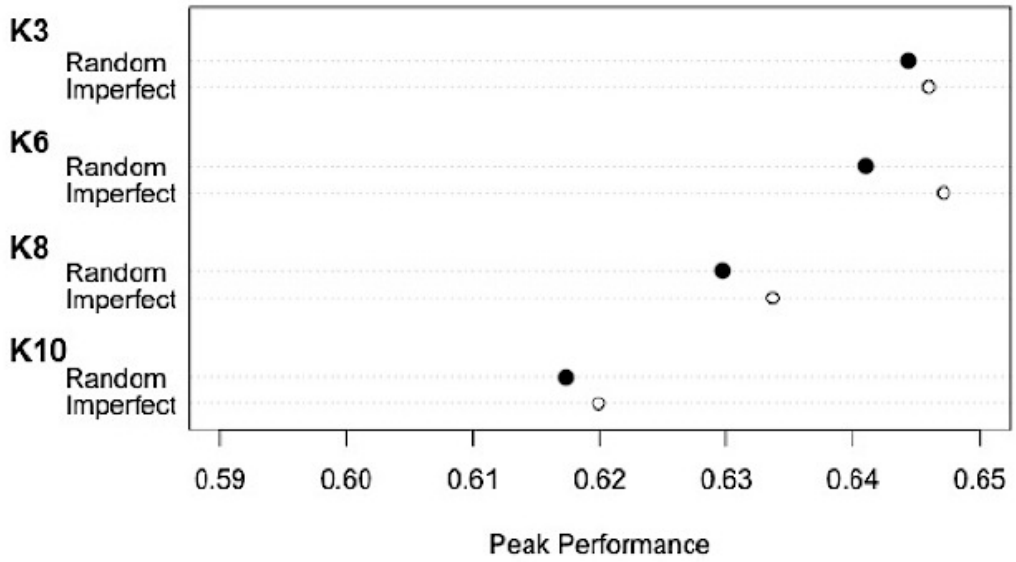
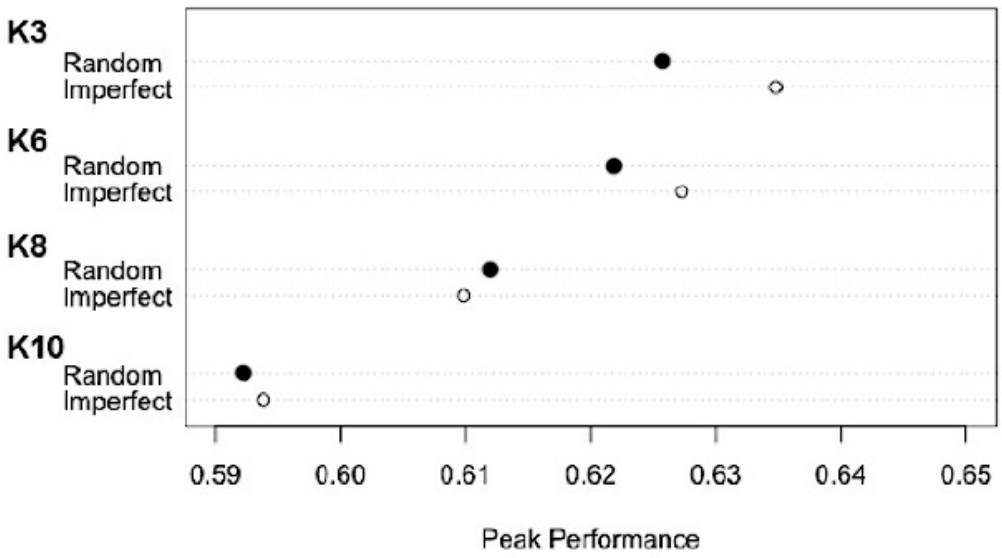


Figure 7(c). Waterfall ISDM with Varied Complexity (N = 16, M = 4, K = 3, 6, 8, 10)



5.2.3. The Effects of Varied Project Size

This experiment takes into account the project size and scope factor while examining the relationship between ISDM and modularity. In this experiment, we vary the number of decision variables *N* and adjust the number of modules *M* accordingly to simulate the size of the project. In this experiment, while we vary *N* of values 12, 16 and 20, we will set $M = 3$ when $N = 12$, $M = 4$ when $N = 16$ and $M = 5$ when $N = 20$. To maintain relative complexities across all experiments, we will specify $K = 8$ when $N = 12$, $K = 10$ when $N = 16$, and $K = 14$ when $N = 20$. As a result, *R* will take the values 96, 160, and 280.

The analyses of the performance relationship between the methodologies and degrees of modularity, taking into account project size and scope, are based on 36 experimental configurations of the ISD project – [(3 x imperfect modular x 3 levels of *N* and *M*) + (3 x non-modular x 3 levels of *N* and *M*)] x 2 ISDM applications = 36 experimental configurations.

Figures 8(a) to 8(d) represents a series of simulations that examine the average performances of the ISD projects using agile and waterfall on the imperfect modular and non-modular designs, where the project size and scope are varied from low to high.

From Figures 8(a) and 8(b), we see that when the agile methodology is used in the imperfect design structure, a project of smaller size and scope (i.e. $N = 12$) yields a much lower performance than projects of larger size and scope (i.e. $N = 16$ and 20). In contrast, when the waterfall methodology is applied, peak performance decreases as project size and scope increases. In addition, the agile methodology outperforms waterfall when $N = 12$ and 16, and yields a similar performance as waterfall when $N = 20$.

While agile methodology works better with modularity than waterfall, the benefits are realized when the project has a decent size and scope. When the size and scope of a project falls below a threshold, the benefits of agile are diminished. However, when the project size and scope gets too large, the benefits of agile over waterfall is diminished. Predictability of a project increases when a project has a smaller size and scope, and decreases when project size and scope is large. As a predictive methodology, waterfall falls in performance when size and scope increases.

Figure 8(a). Agile ISDM & Imperfect Modular Structure with Varied Project Size (N = 12, 16, 20, M = 3, 4, 5, K = 8, 10, 14)

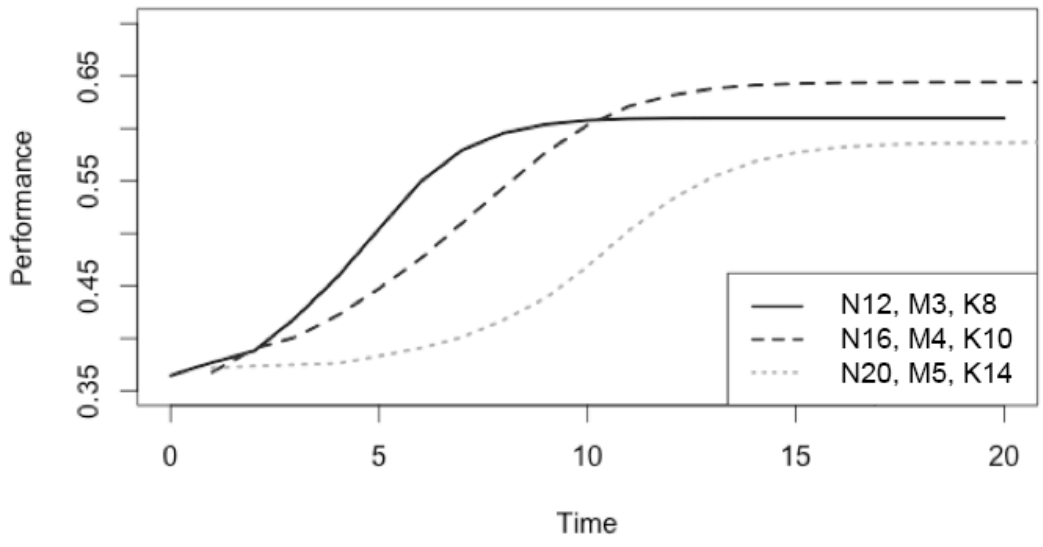
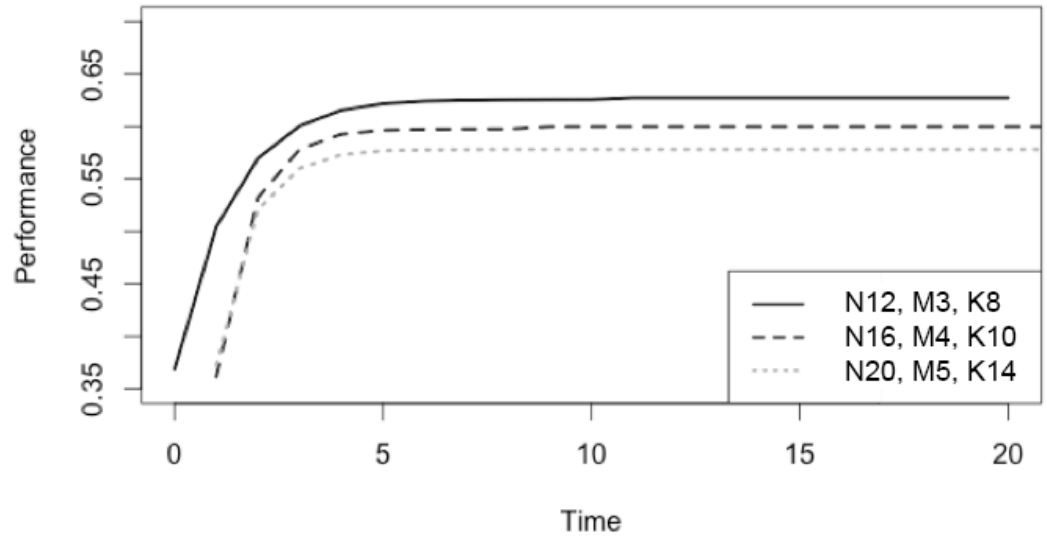


Figure 8(b). Waterfall ISDM & Imperfect Modular Structure with Varied Project Size (N = 12, 16, 20, M = 3, 4, 5, K = 8, 10, 14)



From Figures 7(c) and 7(d), we see that when the project has a non-modular structure, as project size and scope increases, the performance decreases. This is consistent when both agile and waterfall methodologies are applied.

Figure 8(c). Agile ISDM & Non-modular Structure with Varied Project Size (N = 12, 16, 20, M = 3, 4, 5, K = 8, 10, 14)

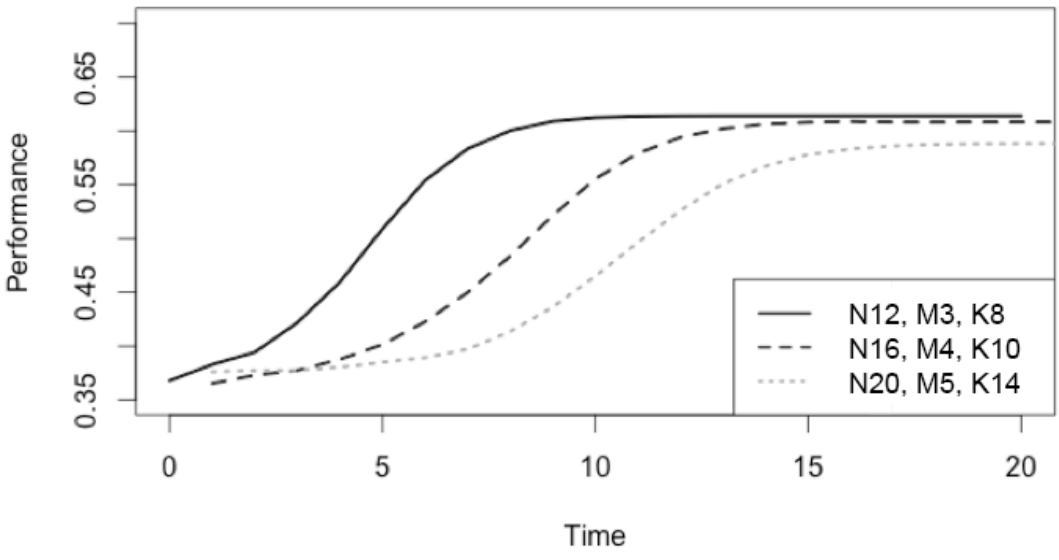
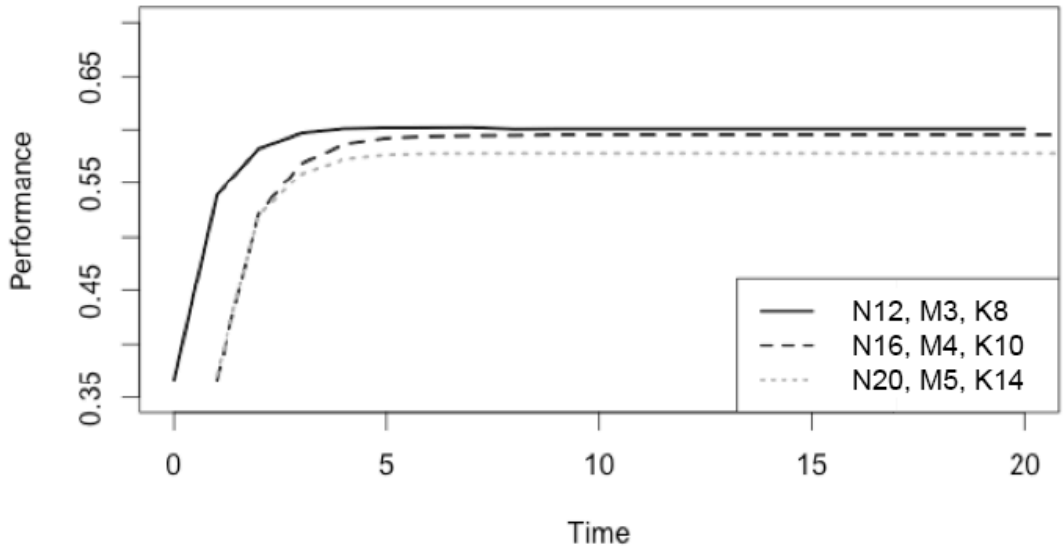


Figure 8(d). Waterfall ISDM & Non-modular Structure with Varied Project Size (N = 12, 16, 20, M = 3, 4, 5, K = 8, 10, 14)



5.2.4. The Effects of Varied Resource Availability

The last experiment in this section considers the effect of the project resources available while examining the relationship between ISDM and modularity. For simplicity of the experiment, project resources will only take into account cost and time budget, as well as manpower. For all simulations in this experiment, we set $N = 16$, $M = 4$, $K = 6$ and $R = 96$. We define three methods to simulate the limited project resources available for the project.

With limited resources (cost, time and effort), the extent of the search of improvement of the project may be impacted. We vary the extent of local search performed by limiting the number of neighboring decision variables to be considered for local incremental improvement.

We limit this boundary to 50%, 75% and 100% of the total available neighboring decision variables, where all neighboring decision variables hold equal chances of being selected for consideration.

The analyses of the performance relationship between the methodologies and degrees of modularity, taking into account project resources availability, are based on 36

experimental configurations of the ISD project – [(3 x imperfect modular x 3 levels of search boundary) + (3 x non-modular x 3 levels of search boundary)] x 2 ISDM applications = 36 experimental configurations.

Figures 9(a) and 9(b) represents a series of simulations that examine the average performances of the ISD projects using agile and waterfall on the imperfect modular and non-modular designs, where the resource constraints are varied from low to high.

From Figures 9(a) and 9(b), it can be noted that with resource constraints, the peak performance of waterfall falls heavily below that of agile in both the imperfect modular and non-modular design structures. When the agile methodology is used, peak performance decreases slightly as resources availability are constrained, however, as the lower the resources availability, the larger the decrease in peak performance of waterfall.

When there are resource constraints, cost, time and effort budget are cut. In the case of waterfall methodology, when there are changes that need to be made later in the project, these changes may not be able to be realized due to resources constraints, as such changes in a waterfall environment are often costly and time-consuming. In contrast, agile promotes flexibility and adaptability throughout the course of the entire project due to modularization. As such, changes are often not as costly as that of waterfall, in terms of cost, time and effort. As such, when resource constraints are high, the impact on the performance of the waterfall methodology is much heavier than that of agile.

Figure 9(a). ISDMs & Imperfect Modular Structure with Resource Constraints

(N = 16, M = 4, K = 6, Resource Availability = 50%, 75%, 100%)

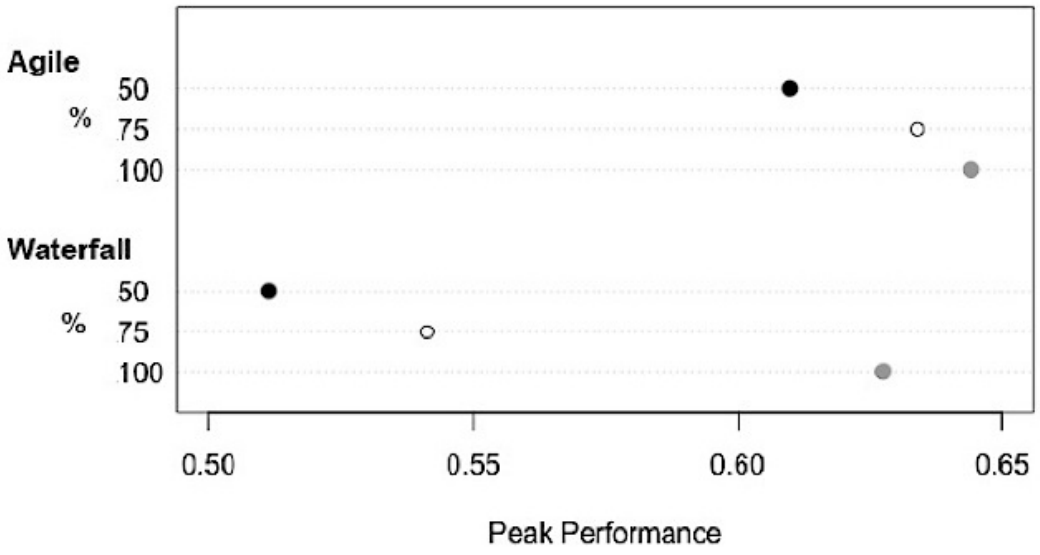
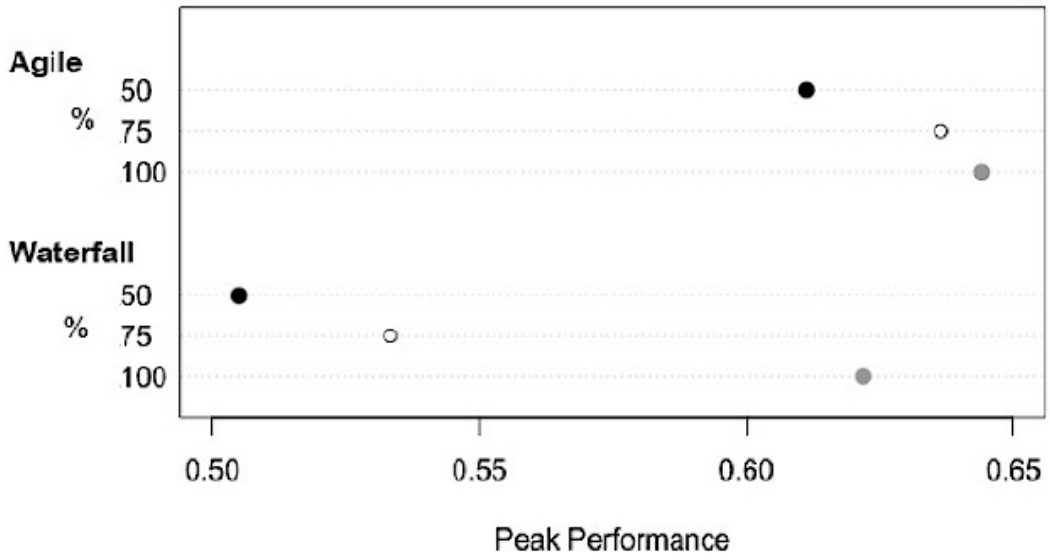


Figure 9(b). ISDMs & Non-modular Structure with Resource Constraints

(N = 16, M = 4, K = 6, Resource Constraint = 50%, 75%, 100%)



5.3. Performance Relationship Between ISDM and Modularity Under Conditions (Multiple Parameters)

To enhance the practicality of the model and experiments, we simulate ISD projects under the conditions where several parameters – uncertainty, complexity, and modularization – are manipulated at once to obtain more insights of performance patterns when multiple parameters are interacting.

5.3.1. Varied Complexity and Uncertainty

In this experiment, we model the uncertainty of real-life ISD projects – such as requirements uncertainty – in conjunction with complexity to simulate a more complicated project environment. The uncertainty parameter holds the values 0% (0.0) to 100% (1.0). Figures 9(a) and 9(b) represents the results of the experiment.

From Figures 10(a) and 10(b), we see that when agile is adopted, it can be noted that the performance for imperfect-modular is better than that of the non-modular structure when complexity and uncertainty is low. As complexity and uncertainty increases by a small amount, performances for all cases are not impacted significantly. However, as complexity and uncertainty increases to higher levels, while performances for both the imperfect-modular and non-modular projects falls in a stable manner when the agile methodology is used, performances drop in a steep manner when the waterfall methodology is used. In essence, the agile methodology suffers when complexity is high, but is resilient to uncertainty, whereas the waterfall methodology is negatively impacted by both parameters.

When project complexity and uncertainty are high, the requirements and technologies involved are complex, thus predictability is low. Due to the predictive nature of waterfall methodology, and also the inability to refactor and make significant changes to the earlier parts of the project, it yields a much lower performance when project complexity and uncertainty are high. In contrast, the adaptive nature of agile methodology allows it to adapt to changing requirements and other project elements through adaptation and refactoring in each iteration throughout the project development. Thus, when complexity and uncertainty are high, agile performs better

than waterfall, despite the possibility of taking a longer time to reach its peak performance.

Figure 10(a). Imperfect-Modular Structure with Varied Complexity and Uncertainty (N = 16, M = 4, K = 3, 6, 8, 10, Uncertainty = 0.0, 0.5, 0.7)

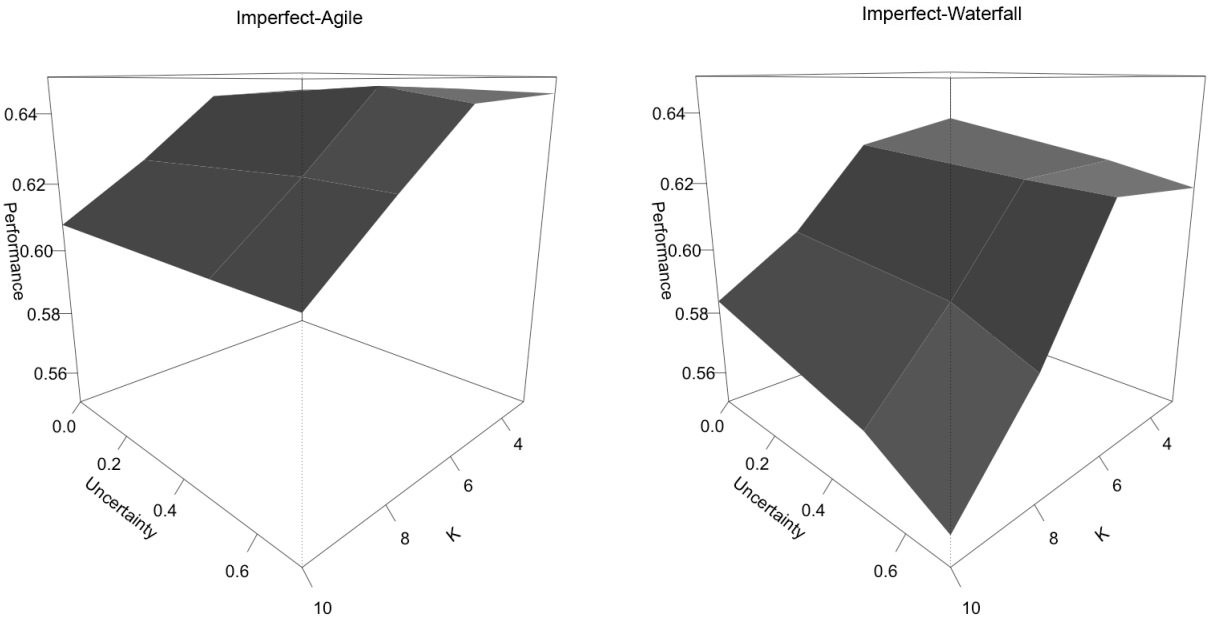
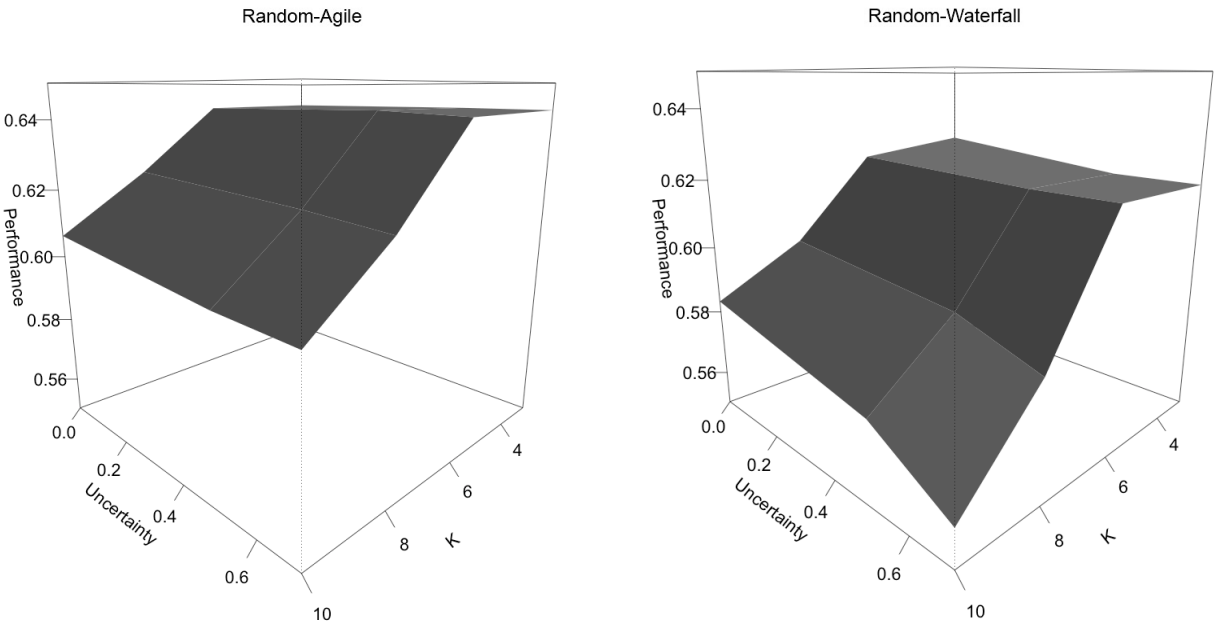


Figure 10(b). Non-Modular Structure with Varied Complexity and Uncertainty (N = 16, M = 4, K = 3, 6, 8, 10, Uncertainty = 0.0, 0.5, 0.7)



5.3.2. Varied Complexity and Modularization

In this experiment, we model the realistic conditions of an ISD project, where complexity varies, and where the ISD team may not know the true underlying structure of the project – thus leading to the possibility of under or over modularization. Figures 11(a) and 11(b) represents the results of the experiment.

From Figures 11(a) and 11(b), we note that assuming that the true underlying structure is $M = 4$, as discussed in section 5.2.1, when the agile methodology is used in the imperfect-modular design, the performance of the true underlying structure outperforms the under and over modularized structures when complexity is relatively low ($K = 3$). As complexity increases, the performance diminishing effects of under and over modularization is more significant, thus the performance of the true underlying structure outperform the other two cases more significantly.

In the case of under-modularization, as complexity increases, the number of decision variables increases in each module as shown in Table 1 below. Thus, the number of decision variables to be considered in each iteration increases as complexity increases.

Table 1. Number of Decision Variables per Module with increasing K

K	Number of decision variables in each module
3	2
6	4
8	7
10	7

In the case of over-modularization, as discussed in section 5.2.1, there are too few decision variables to be considered in each iteration, thereby diminishing the performance attainable for both the imperfect-modular and non-modular projects. This effect is amplified as complexity increases.

When the agile methodology is applied on the non-modular structure, the performance-weakening effects of both under and over modularization are not as apparent as that of the imperfect-modular structure, while performances still decreases to some extent. However, consistent with the latter, performance of the true underlying structure outperforms. Comparing the performances of the imperfect modular and non-modular projects when the waterfall methodology is applied, the performance differences when $M = 2, 4$ and 6 does not differ significantly. As the waterfall methodology considers all decision variables in each iteration, the number of modules contributes a minimal effect on the performance attainable. However, as complexity increases, the performance of both the imperfect and non-modular structures decreases consistently. It can also be noted that while the performance of the imperfect modular structure outperforms that of the non-modular structure when the waterfall methodology is applied, the performance gap closes up as complexity increases because the performance of the imperfect modular structure decreases slightly steeper than that of the non-modular structure.

Figure 11(a). Imperfect-Modular Structure with Varied Complexity and Modularization (N = 16, M = 2, 4, 8, K = 3, 6, 8, 10)

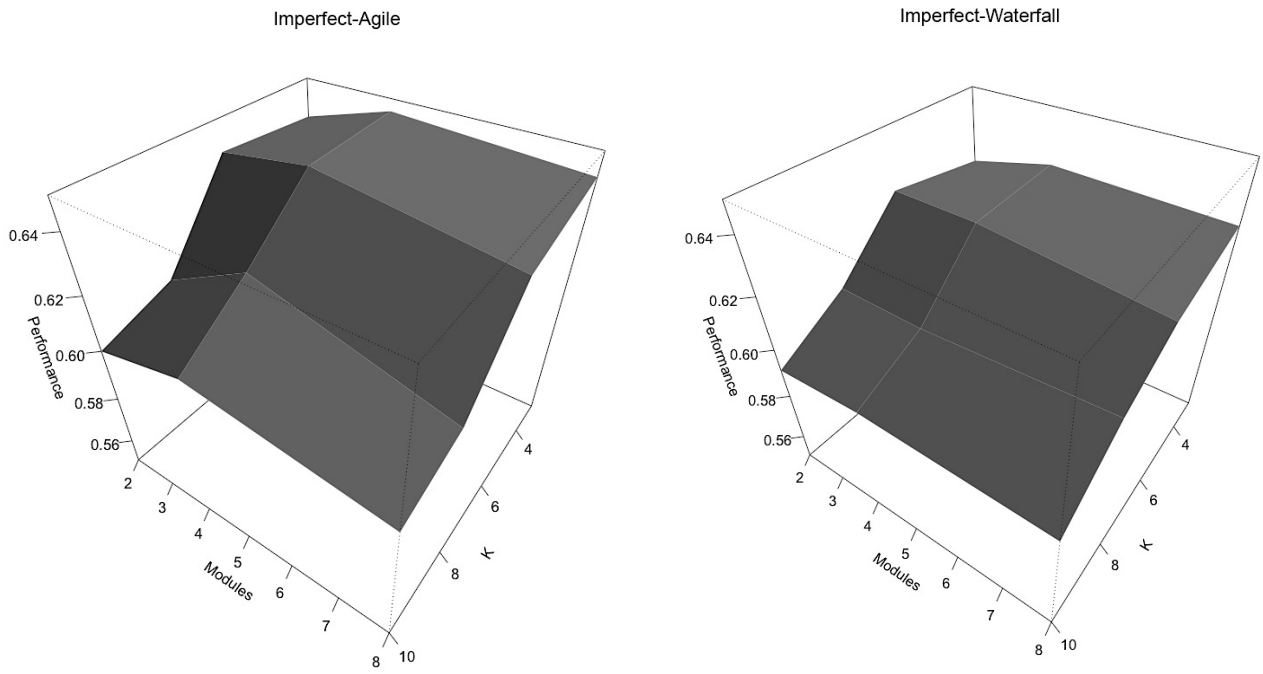
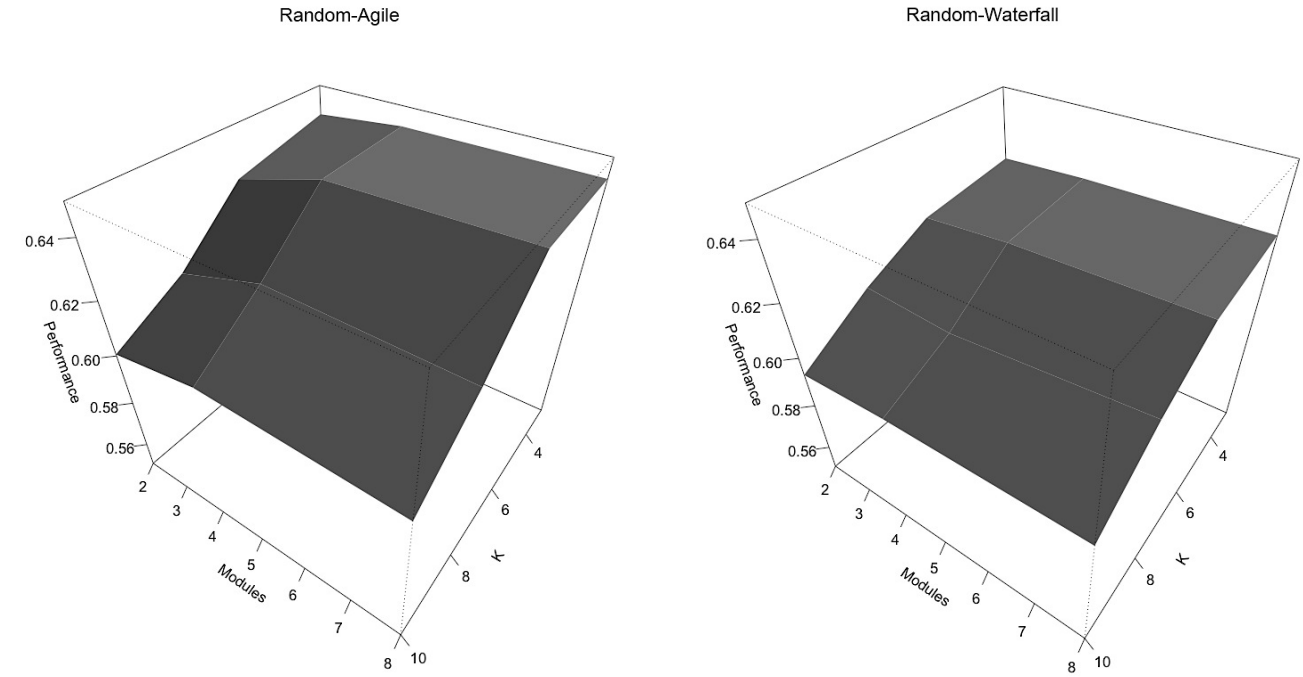


Figure 11(b). Non-Modular Structure with Varied Complexity and Modularization (N = 16, M = 2, 4, 8, K = 3, 6, 8, 10)



6. Discussion

Existing research on the selection of ISDM for projects are largely revolved around project characteristics such as project size, complexity and organizational structure. Minimal emphasis has been given to the problem modularity of projects. Using the *NK* fitness landscape model, we explored the impact of various levels of modularity on the application of agile and waterfall methodologies. We also examined the impact of project elements such as complexity, uncertainty, size and scope, as well as resource availability on the project performance with various degrees of modularity. We aimed to provide some insights to the question: Under various environmental factors, which ISDM should an ISD team adopt under which degree of problem modularity?

6.1. Summary of Results

When the agile methodology is used, the imperfect modular structure performs better than that of the non-modular structure in all cases. However, the perfectly modular structure underperforms significantly. When the waterfall methodology is used, the performances of the imperfect and non-modular structures are relatively consistent, with the perfectly modular structure underperforming. Further, the effects of under and over modularization are only apparent in the projects adopting the agile methodology, where both under and over modularization are seen to deal negative effects on performance. The effect of increasing complexity as well as uncertainty decreases performances of projects adopting both the agile and waterfall methodology, with greater impact on waterfall projects. On the other hand, the effect of increasing project size decreases waterfall project performance consistently, whereas agile projects of relatively small and relatively large sizes underperform. When resource availability is limited, the impact on the performances of waterfall projects is significantly greater than that of agile projects. In all cases, the imperfectly modular projects are more resilient than non-modular projects when the agile methodology is used.

6.2. Implications of Results

To answer the question that we posed, when a project problem is adequately modular, of a decent size and scope, the project team can attain the highest possible performance using the agile methodology. However, as discussed in section 5.1, a fully modular problem structure and a small project scope will instead diminish the benefits of agile.

As supported by previous research done by Rivkin and Siggelkow, in a perfect modular structure, the interdependencies are fully clustered within modules. As the interdependencies within a problem clusters tightly into modules, the number of local peaks increases significantly. Thus, this reduces the possibility of attaining the highest performance possible, which explains the low performance of a perfect modular structure. In contrast, a certain extent of modularity – a level between being perfectly modular and non-modular – realizes the benefits of the agile methodology, being adaptable and flexible. Improvements can be made to modules developed before, to achieve an overall higher quality. Referring back to the example of the development of a sales and marketing system discussed in the model earlier, in a particular ISD project, when a module is set to the enable customer order tracking, another module that is set to enable order tracking through push notifications will be dependent on the customer order tracking module. Thus, performance improvements can be made to both modules throughout the iterations to ensure the coupling of modules is optimized, thereby giving rise to higher product quality that enables more functions accessible and interoperable by customers. However, consider the case of the ISD project consist of a module that is set to enable customer order tracking, and another module that enables the procurement of supplies that is independent of the customer order tracking module. Once the modules are developed and a local peak performance is achieved, further improvements will not be made due to the independence of both modules. In addition, project teams must also be cautious as to whether or not the ISD project is under or over modularized. When a project is under-modularized, the number of decision variables to consider at each iteration increases, and decreases when the project is over-modularized. As seen in Figure 6(a), myopically, as the number of decision variable to be considered in each iteration increases, the room of performance improvement increases, thus there is a higher possibility of attaining a greater peak performance. However, as the number of decision variables to be considered in each iteration increases, the number of local peaks increases, and it places more effort on the ISD team to discover the next performance peak[9]. However, as previously discussed in section 5.2.3, to fully realize the benefits that agile methodology brings, the ISD project needs to be of a decent size and scope. The adaptable and flexible benefits of the agile methodology are not realized by projects of very small size and scope. In contrast, when projects are of small size

[9] As discussed in section 5.1, as the structure places more dependencies within each module, the number of local peaks of performances increases, making it harder to attain better performances.

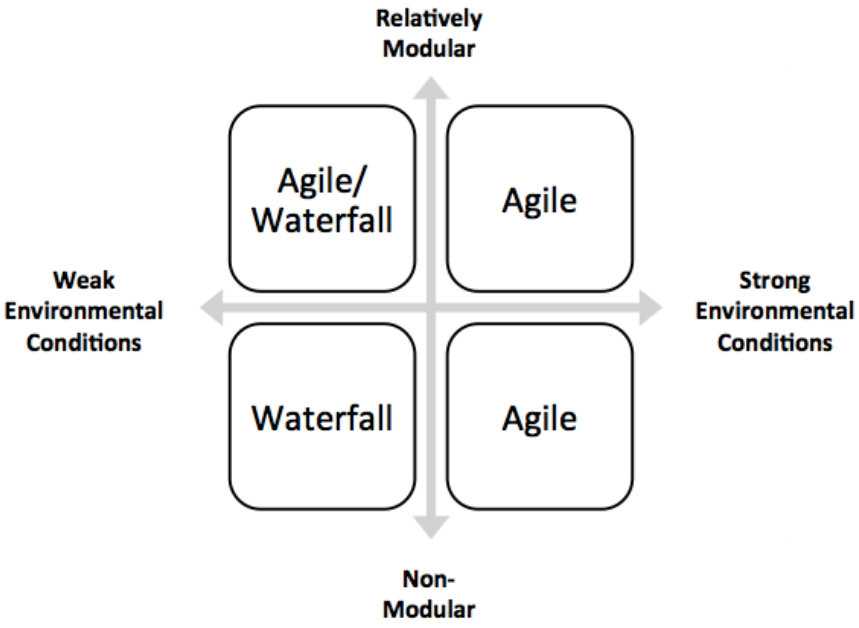
and scope, the predictability and uncertainty of the problem lessens. Thus, the predictable nature of the waterfall methodology suits such project cases better than the agile methodology.

Referring to sections 5.2.4 and 5.3.1, the agile methodology is significantly more resilient to environmental factors – project uncertainty, complexity and resource constraints – as compared to the waterfall methodology. However, when environmental factors are strong – which causes the project landscape to be rugged – the performance difference between the imperfect and non-modular structures shrinks. For projects of both imperfect and non-modular designs, the agile methodology is significantly more resilient to higher project uncertainty, complexity as well as higher resource constraints than the waterfall methodology. When environmental factors are strong, the project uncertainty is higher, as well as complexity. The predictable nature of the waterfall methodology works by considering all decision variables in each iteration, thereby requiring significantly more effort to achieve or maintain performance as compared to agile. In addition, as environmental factors become stronger, the predictability of the problems decreases significantly, defying the nature of the waterfall methodology. Consequently, as the environmental factors become stronger, the waterfall projects achieve a much lower performance yield. In contrast, the agile methodology simulates a "divide-and-conquer" strategy when developing ISD projects, by breaking down a large and complex problem into smaller chunks – in this case, modules – and completing the work on one module before moving on to the next. Agile methodology also promotes adaptability, where changes can be made to modules that are completed to adapt improve performance. This strategy reduces the effort needed to maintain the project, facilitates changes and improves the overall project quality as compared to the waterfall methodology.

In essence, the points discussed above can be summarized into the matrix shown in Figure 12 below. When a project team faces strong environmental factors (large scope, high uncertainty, high complexity), the resilience of agile makes it a more favorable ISDM to adopt, regardless of the modularity of the project design as the performance difference between imperfect and non-modular designs shrinks with strong environmental factors. However, when the team faces weak environmental factors (small scope, low uncertainty, low complexity), the project is rather predictable, and the

waterfall methodology works well when the project structure is non-modular. In the case where the project structure is relatively modular with weak environmental factors, the performance gain from using agile is not significantly more than that of the use of the waterfall methodology. Thus in this case, both the agile and waterfall ISDMs are adequate.

Figure 12. Modularity and ISDM Matrix with Environmental Factors



In managerial terms, to attain the highest performance possible by using the agile methodology, it is crucial for ISD teams to ensure that they achieve an adequately modular and sized project at the initial starting point. In reality, this accuracy will not be achieved in one project. The ISD team will have to go through several times of project planning and development to obtain the experience needed to determine how the project elements should be structured best for the use of agile. Planning is always imperative for project management. This planning phase should be structured such that the ISD team explore the various requirements of the ISD project and structure them accordingly in modules, and benchmark the results accordingly with past experiences on whether the planned modularity and size of the project is adequate for agile to be used.

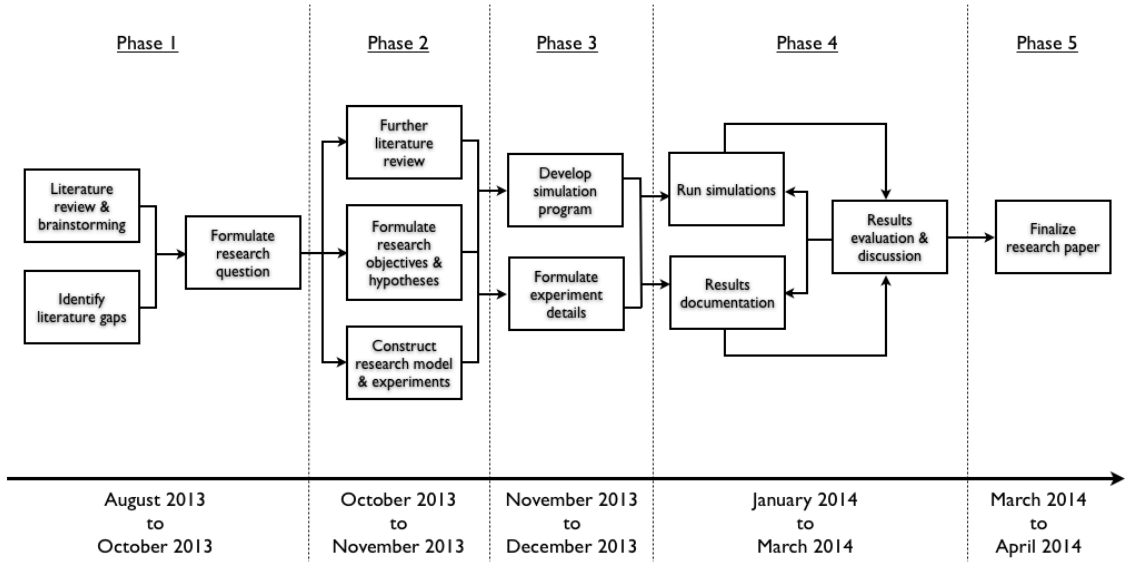
6.3. Limitations and Future Research

Nonetheless, in this paper, we have only looked at the fundamental performance relationship between modularity and the agile and waterfall ISDMs alongside with several environmental parameters. There is still a large room for further research on this link. While the agile methodology is simulated to be of a "divide-and-conquer" strategy in this paper, research has shown that other forms of agile can be applied to ISD projects of large scope and complexity, such as the "divide-after-conquer" strategy (Elshamy & Elssamadisy, 2006). This development practice involves solving the base problem first with a smaller development team (the conquer phase), before expanding the project team to its full size (the divide phase). This agile practice is anticipated by some researches to solve many of the problems that occur with larger projects using agile methodologies. To find out more about modularity and its performance relationship with the agile methodology, our experiments in this paper can be extended to such forms of agile for further research.

7. Project Management

The critical path analysis flow diagram shown in Figure 13 shows the breakdown of activities undertaken through the course of this project. Phase 1 consists of the exploratory activities and phase 2 consists of the initial start of research formulation. Phases 3 and 4 consist of the development of the simulation program, the running of experiments, as well as the evaluation of results. Phase 4 is a looping phase where simulations are run and results are documented and evaluated. Lastly, phase 5 is a short phase for the finalizing of the research paper. In essence, the timeline was accurate, and the bulk of effort was concentrated on phase 3 and 4.

Figure 13. Critical Path Analysis Flow



In terms of resource management, we expected to utilize the High Performance Computing (HPC) resources for the running of experiments and simulations. However, we were able to utilize the School of Computing's Computer Cluster (Tembusu and Angsana Clusters) to run the batch jobs in phase 4. This greatly facilitated convenience by allowing the running of batch jobs remotely.

During the course of this project, most challenges arose in phases 3 and 4. Since the simulation program has to be tweaked to fit the purpose of the experiments in this paper, there were some technical complications. However, with guidance by the

supervisor as well as exploration, we were able to solve the technical problems quickly. Also, unfortunately some results from our initial experiments did not reveal many insights. Thus, we refined the experimental settings and expanded the scope of this research to gather more insights through various experiments, as presented in this paper.

References

Amaral, L. A., & Uzzi, B. (2007). Complex Systems—A New Paradigm for the Integrative Study of Management, Physical, and Technological Systems. *Journal of Management Science, 53* (7), 1033-1035.

Baldwin, C. Y., & Clark, K. B. (2000). *Design Rules. Vol. I: The Power of Modularity*. The MIT Press, Cambridge, MA.

Baldwin, C. Y., & Clark, K. B. (2004). *Modularity in the Design of Complex Engineering Systems*. Harvard Business School, Boston, MA.

Baldwin, C., & Clark, K. (2001). *The value and costs of modularity*. Harvard Business School, Boston, MA.

Brooks, F. (1975). *The Mythical Man-Month: Essays on Software Engineering*. Addison-Wesley, MA.

Brusoni, S., Marengo, L., Prencipe, A., & Valente, M. (2007). The value and costs of modularity: a problem -solving perspective. *European Management Review, 4* (2), 121-132.

Cerveny, R. P., Garrity, E. J., & Sanders, G. L. (1990). A problem-solving perspective on systems development. *Journal of Management Information Systems, 6* (4), 103-122.

Conley, C. A., & Sproull, L. (2009). Easier Said than Done: An Empirical Investigation of Software Design and Quality in Open Source Software Development. *42nd Hawaii International Conference on System Sciences*. Jan 6-9, Waikoloa Village, HI.

Davis, J., Bringham, C., & Eisenhardt, K. (2007). Developing Theory Through Simulation Methods. *Academy of Management Review, 32* (2), 480-499.

Duffy, A., & Ferns, A. F. (1998). An analysis of design reuse benefits. *Proceedings of the 12th International Conference on Engineering Design* (pp. 799-804). Design Society. Aug 24-26, Munich.

Duffy, A., Smith, J. S., & Duffy, S. M. (1998). Design reuse research: a computational perspective. *Engineering Design Conference on Design Reuse*. Jun 23-25, London, UK.

Dyba, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology, 50* (9-10), 833-859.

Elshamy, A., & Elssamadisy, A. (2006). Divide After You Conquer: An Agile Software Development Practice for Large Projects. *7th International Conference, XP 2006. 4044*, pp. 164-168. Oulu, Finland: Springer.

Ethiraj, S. K., Levinthal, D., & Roy, R. R. (2008). The Dual Role of Modularity: Innovation and Imitation. *Management Science, 54* (5), 939-955.

Geambasu, C. V., Jianu, I., Jianu, I., & Gavrila, A. (2011). Influence factors for the choice of a software development methodology. *Accounting and Management Information Systems , 10*(4), 479-494.

Hahn, J., & Lee, G. (2011, September). Shared Domain Knowledge in Information Systems Development: An NK Fitness Landscapes Model, Working Paper, NUS, Singapore.

Kässi, T., Leisti, S., & Puheloinen, T. (2008). Impact of product modular design on agile manufacturing. *Mechanika, 74* (6), 56 -62.

Kauffman, S. A. (1993). *The origins of order: Self-organizing and selection in evolution*. New York: Oxford University Press.

Kauffman, S. A., & Weinberger, E. D. (1989). The NK model of rugged fitness landscapes and its application to maturation of the immune response. *Journal of Theoretical Biology, 141* (2), 211-245.

Korpela, M., Mursu, A., & Soriyan, H. (2002). Information Systems Development as an Activity. *Computer Supported Cooperative Work, 11* (1-2), 111-128, Kuopio, Finland.

Langlois, R. N. (2002). Modularity in technology and organization. *Journal of Economic Behavior & Organization, 49* (1), 19-37.

Levinthal, D. A. (1997). Adaptation on Rugged Landscapes. *Management Science, 43* (7), 934-950, Philadelphia.

Narduzzo, A., & Rossi, A. (2003). Modular Design and the Development of Complex Artifacts: Lessons from Free/Open Source Software. *ROCK Working Papers , 21*. Italy.

Oracle. (2007). *The Benefits of Modular Programming*. Sun Microsystems.

Phariss, R. (2006, November 29). The Importance of Requirements Definition in IT Systems Development.

Simon, H. A. (1996). *The Sciences of the Artificial*. The MIT Press.

Rivkin, J. W., & Siggelkow, N. (2007). Patterned Interactions in Complex Systems: Implications for Exploration . *Management Science , 53* (7), 1068-1085.

Ulrich, K., & Eppinger, S. (1999). *Product Design and Development* (2nd Edition ed.). New York: McGraw-Hill.

Vavpotič, D., & Vasilecas, O. (2012). Selecting a Methodology for Business Information Systems Development: Decision Model and Tool Support. *Computer Science and Information Systems , 9* (1), 135-164.

Xia, W., & Lee, G. (2005). Complexity of Information Systems Development Projects: Conceptualization and Measurement Development. *Journal of Management Information Systems, 22*(1), 45-83, Minnesota.

Yan, X. T., Stewart, B., Wang, W., Tramsheck, R., Liggat, J., Duffy, A. H., et al. (2007). Developing and applying an integrated modular design methodology within a SME. *International Conference on Engineering Design*. Paris: Cite Des Sciences Et De L'Industrie.

Appendix 1 – Adequacy of Experiential Settings

The experiment was set with 100 independently generated fitness landscapes for each influence matrix, with 50 ISD projects randomly seeded onto each landscape. However, results have shown that the same experiment set with 50 landscapes and 20 ISD projects yield approximately the same results with negligible differences in the standard deviations of performance at each time period, as shown in the tables below. Hence, we conclude that the setting of 50 landscapes and 20 agents is appropriate for our experiments.

For each table:

- The columns "A", "B" and "C" show the average standard deviations of performance across 20 time periods.
- The columns "D" and "E" show the difference between the standard deviations in columns "A", "B" and "C".

Time Period	(A) 50 Landscapes 50 Agents	(B) 100 Landscapes 20 Agents	(C) 50 Landscapes 20 Agents	(D) A&C Difference	(E) B&C Difference
Perfectly Modular (Agile)	0.073244202	0.072526634	0.073677053	0.00062639	0.000626609
Perfectly Modular (Waterfall)	0.066526184	0.066044624	0.066581969	0.000281958	0.000543834
Imperfect Modular (Agile)	0.071520802	0.069021867	0.071126881	0.000547598	0.004597767
Imperfect Modular (Waterfall)	0.063511724	0.061398446	0.063731787	0.000434316	0.000648737
Non-Modular (Agile)	0.069618497	0.070187623	0.069559318	0.000614461	0.0006964
Non-Modular (Waterfall)	0.0006964	0.062429381	0.058543636	0.000321063	0.000527303

End

of

Paper