Control Mechanisms for Managing Modularized ISD projects

Completed Research Paper

Subasinghage Maduka Nuwangi **Darshana Sedera**

Shirish C. Srivastava

Abstract

Information Systems Development (ISD) projects widely utilize modularization to enable better management and control. Yet, the modalities for effectively managing modularized ISD projects are not clearly established. By adopting a 'control theory' perspective and leveraging case study approach, we examine eight modularized ISD projects to unearth the mechanisms for managing modularized ISD projects effectively. Results demonstrate that lack of module interdependencies increase the use of formal controls and decrease the informal clan controls. However, informal clan controls may be required for team members to understand the project requirements. Further, it was found that an error in the identification of module interdependencies creates fluctuations in Business Requirement Specifications (BRSs), which in turn, originates project management issues at the later stages of the projects. Rather than utilizing the component-sharing modularity, the use of sectional modularity minimizes the fluctuations in BRS, which ultimately reduces the project management issues in ISD projects.

Keywords: Information Systems Development, Modularization, Control, Interdependencies

Introduction

Modularization is a technique widely employed in information systems development (ISD) projects to minimize cost (Dedrick et al. 2011), reduce complexity (Gershenson et al. 2003) and increase the ability to maintain and manage the projects (Ravishankar and Pan 2013). Modularization has been defined as the process of "decomposition of complex tasks into simpler portions so they can be managed independently and yet operate together as a whole" (Mikkola and Skjøtt-Larsen 2004, p. 354). The simpler portions of the complex tasks are referred to as modules. Quality of the decomposition of software modules are measured by the level of coupling and cohesion (Adamo-Villani et al. 2009). Coupling of a module characterizes its interdependencies with other modules, whereas cohesion of a module characterizes its internal interdependencies (Allen and Khoshgoftaar 1999). Several researchers (e.g., Allen and Khoshgoftaar 1999; Goulão 2001) highlighted the importance of planning for low coupling in modularization. When software projects include low coupling, it minimizes the interdependencies between the modules, thereby enabling efficient control (Sanchez 1995). However, with the growing complexity of software functionalities, ISD projects are increasingly having interdependent modules requiring a clear focus on managing these external dependencies or couplings.

Our study is motivated by this emergent need and examines the mechanisms through which ISD projects with interdependent modules can be managed and controlled.

Modularization is conducted in almost every stage of an ISD project including the requirement analysis, design, coding and testing stages (Al-Otaiby et al. 2005; Nuwangi 2016). The current study focuses on the requirement analysis stage of an ISD project. In the requirement analysis stage, client requirements are decomposed into simpler portions, which are referred to as requirement modules. Although low coupling (i.e. lack of interdependencies among software modules) is considered as the main attribute related to the quality of the decomposition of modules, requirement modules in ISD projects tend to have interdependencies. Figure 1 conceptually represents the nature of requirement modules in ISD projects. Since module B is interdependent on requirement information specified in module A; an update in module A can cause changes in module B. As depicted in Figure 1, modules A and C are overlapping. As a result of these overlaps, information in module C has to be updated in conjunction with the updates in module A and vice versa.

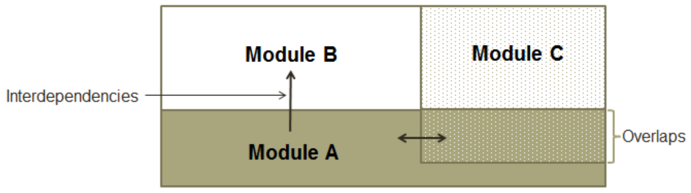

Figure 1: Interdependencies between modules

Formal and informal controls suggested by the control theory literature (Kirsch 1996; Rustagi et al. 2008) can be used to manage projects with projects with interdependent modules. Formal control involves controlling the employees through performance evaluation in which either the outcomes or behaviors of the employees are measured, evaluated and rewarded (Kirsch 1996). Informal control uses social or people strategies to control the employees. When the projects include modules with several interdependencies, the projects require intensive controls because a change in one module may affect the other interrelated modules (Sanchez and Mahoney 1996). Modules with less interdependencies provide 'embedded coordination' that minimizes the need for continuous management supervision (Sanchez 1995). According to Sosa et al. (2004), modularization creates boundaries between different teams, thereby increasing communication barriers. As a result, the teams in modularized projects tend to work independently and have little communication with the other teams. Thus, project managers should enforce appropriate controls to ensure that the tasks completed by a team are aligned with the other teams' tasks. Although modularization and project control are interrelated (Sosa et al. 2004), ISD literature rarely explores the linkages between modularization and control. Given the theoretical and practical salience of examining these linkages, the present study views the management of ISD projects from a control theory perspective and addresses the research question: "What are the control mechanisms for managing modularized ISD projects effectively?"

In ISD projects, client requirements grouped under every module are described using a business requirement specification (BRS)[1]. Due to interdependencies between modules, BRSs may also include interdependencies. During ISD project implementation, BRSs can be updated as a result of volatile client requirements (Dingsøyr et al. 2012) and ISD team members' suggestions. Scope creep (i.e. not understanding the size and complexity of tasks) would also lead to BRS updates (Schmidt et al. 2001). When BRSs are interdependent, updating a BRS can cause changes in many other BRSs. Thus, software engineers may have to update several parts of the software code, as a result of changing a single line in BRS, which describes a single aspect of a business requirement. Likewise, interdependencies between modules increase the complexity of ISD project control.

By understanding the nuances associated with modularization and control, the current study contributes significantly to the control theory literature concerning ISD. In particular, the outcomes of the study will facilitate a better understanding of the nature of modularization in contemporary ISD projects and

[1] BRS comprises - software requirement specifications, functional specifications, product specifications, system specifications and requirement documents.

provide insights to both researchers and practitioners for improving the effectiveness of modularized ISD project control.

Theoretical Background

Although minimum interdependencies between modules are preferred (Goulão 2001), modules in ISD projects tend to have interdependencies, thereby increasing the complexity of ISD project control. Since the control theory (Eisenhardt 1985; Kirsch 1996) provides a theoretical lens that supports for understanding and resolving issues in complex ISD projects (Nuwangi et al. 2014; Sedera et al. 2014), the current study is grounded in this perspective. The control theory literature (Kirsch 1996; Rustagi et al. 2008) explains formal and informal controls. Formal control can be subdivided into outcome-based and behavior-based control modes. The outcome-based mode is implemented through mechanisms that specify the expected outcomes, whereas the behavior-based mode is implemented using the mechanisms influencing appropriate behaviors. Companies utilize variety of techniques such as BRSs and timelines to inform the team members about the expected outcomes and behaviors of the projects (Choudhury and Sabherwal 2003; Nuwangi et al. 2018). In modularized ISD projects, team members are assigned to different modules and are provided with relevant BRSs. According to Jaworski (1988, p. 27), informal control mechanisms are "unwritten, typically worker-based mechanisms that influence individual or group behavior." The informal control consists of clan and self-control modes. Ouchi (1978) discusses clan control as promoting common values and beliefs within a clan, which is defined as a group of individuals who share a set of common goals. In contrast, self-control occurs when the employees of the company control their own actions (Manz and Angle 1986). As a result of assigning team members to different modules, ISD projects consist of multiple interdependent clans.

Pine (1993) identifies different types of modularization: 1) component-sharing modularity - includes sharing the same component across multiple products, 2) component-swapping modularity - "different components paired with the same basic product, creating as many products as there are components to swap" (Pine 1993, p. 202), 3) cut-to-fit modularity - similar to the previous two types of modularity, except that in cut-to-fit modularity one or more of the components is variable within practical limits. This is suitable for products when some components can be continually changed to manage different product requirements, 5) bus modularity - provides the standard structure in which many components can be attached together, and 6) sectional modularity - allows the configuration of any number of components, until each component is connected with other components with a standard interface.

Two gaps in the prior research on modularization and control theory are particularly noteworthy. First, the literature rarely discusses the linkages between modularization and control (Tiwana 2008). Tiwana (2008) states that modularity substitutes control, but does not describe the mechanisms for controlling modularized ISD projects effectively. Furthermore, Tiwana (2008) focuses solely on formal controls and does not explicitly discuss the relationship between modularization and informal-clan control. Second, the modularization literature states that modularization minimizes the need for organizational control. For example, Ron (1995) argues that modularization provides embedded coordination by minimizing the need to interact with team members from other modules. However, interdependencies between modules create the need of interactions between team members from different modules. Furthermore, projects may need control due to other issues such as the paucity of information specified in the BRSs. A theoretical explanation for this remains absent. Therefore, to address this gap, the present study explores this uncovered, yet essential, perspective of control mechanisms for managing modularized ISD projects effectively.

Theoretical Propositions

In ISD projects, each team member is provided with relevant BRSs, which include details about the modules. According to Haag et al. (1996), customer requirements might be included in multiple BRSs. Therefore, BRSs which are assigned to different team members can be interdependent. Cataldo et al. (2008) highlight that when team members coordinate their tasks with other team members, the quality of work increases. As per Chou et al. (2011), software products with a large number of interdependent tasks are difficult to implement. Therefore, ISD projects utilize several techniques such as modularization to minimize the task interdependencies (Clark and Baldwin 2000). Since there can be unanticipated interdependencies between software development tasks (Kraut and Streeter 1995), modularization can be challenging. When information systems functionalities are modularized, the team members can be provided with tasks that are less interdependent. This provides the ability for the project managers to measure the outcomes and behaviors of the team members effectively. Therefore, when projects include modules with fewer interdependencies, project managers tend to utilize formal controls for governing the team members. Hence, it is proposed:

P1: When the level of interdependencies between the modules is low, it is likely that the projects will execute formal controls

Pflügler et al. (2018) discussed the formation of subgroups in ISD projects. ISD projects consist of subgroups such as the business analyst team, software engineering team and software quality assurance team. Although these subgroups are significantly different in their formal structures, and departmental missions, all subgroups share the same project goals and objectives (Ouchi 1980). For example, while business analysts are required to identify the business requirements and write the BRS, software engineers are required to develop the software according to the BRS. Hoegl et al. (2003) state that the lack of collaboration between team leaders can cause the duplication of tasks, thereby reducing the ability to complete the project within the stipulated time and budget. Frequent interactions between team members provide the ability to build a common understanding of the project goals. In modularized ISD projects, the team members are assigned to different modules. For example, a financial management module consists of members from the software engineering team and software quality assurance team. According to Sosa et al. (2004), modularization creates boundaries between teams, thereby increasing communication barriers. As a result of lack of interdependencies between modules, teams are able to complete the tasks independently. This minimizes the use of informal-clan controls in projects. Therefore, it is proposed:

P2: When the level of interdependencies between the modules is low, it is unlikely that the projects will execute informal-clan controls

Research Methodology

In order to scrutinize the theoretical propositions, the study takes a case study approach. The case study approach has been recognized to be appropriate for research that explores complex environments (Klein and Myers 1999) and contemporary events (Benbasat et al. 1987). The case study follows a deduction logic through which it is intended to provide valuable insights into pre-identified propositions (Lee 1989). The qualitative case study approach enables the propositions to be examined in an exploratory manner, thereby increasing the richness of the findings (Dibbern et al. 2008). The selection of the sample in the study followed the purposeful sampling strategy in line with the research objectives (Patton 2002). Three conditions formed the benchmarking criteria for the selection of the ISD organization. First, the organization should modularize the business requirements. Second, the organization should be involved in multiple ISD projects to provide flexibility in data collection. Third, the organization must be sufficiently large, with a standard hierarchy of employment. This enables collecting data from team members at different levels of employment. Following the application of these criteria, Company A was selected as the case organization. Company A was a medium sized ISD company, engaging in stock market-related ISD. The company specializes in developing IS solutions for capital markets, with more than 25 capital market clients all over the world. Those solutions provide the ability to trade using multiple assets and manage securities lending and borrowing[2]. Eight (8) ISD projects (see Appendix A for the project descriptions) within Company A were selected as the cases to ensure control and replicability. The selection of the projects followed the opportunistic/emergent sampling strategy. The selected projects were similar in terms of the industry sector (ISD projects), type of information systems developed (stock exchange systems), project stage (completed), but varied in terms of clients (from different countries), team members (different personnel), and outcomes

[2] For a wider discussion of the above concepts see for example Senarath and Copp (2015), Senarath (2016) and Senarath (2017)

(success/failure). The selected projects were already completed; this enabled the respondents to discuss the control mechanisms of the entire project.

Data Collection and Analysis

Twenty-three (23) semi-structured interviews, each lasting 25–30 minutes, were conducted with employees from the selected eight (8) ISD projects. See appendix B for list of interview questions. In order to avoid key informant bias, interviews were conducted with multiple informants from each project (Kumar et al. 1993). ISD projects generally consist of a business analyst team, software engineering team, quality assurance team and a project management team. While the business analysts are responsible for documenting the BRS, software engineering team is responsible for developing ISD solutions according to the BRS. Responsibilities of software quality assurance team include testing the ISD solutions to identify non-compliance issues. Project management team is responsible for ensuring that the project is executed according to the project plan. A project manager, a team member from the business analyst team (business analyst, senior business analyst, consultant or senior consultant) and a team member from the software engineering team (software engineer, senior software engineer or technical lead) were interviewed (see Appendix C for participant information)[3]. The sampling was non-probabilistic, purposive and employed the 'snowball' technique, whereby the informants were appropriate opinion leaders with well-developed views on the research topic (Minichiello et al. 1995).

All the interviews were audio-recorded and transcribed for subsequent data analysis. The interview data was supplemented with documents such as BRSs, test scenarios and test case specifications. These documents increased the validity and reliability of the collected data. ISD project was the unit of analysis. NVivo software was used for data analysis. The data analysis process was commenced with a deep understanding of theoretical domain of modularization and ISD project controls. Since this research follows a deduction logic, coding guideline was created including initial characteristics of each concept (i.e. level of interdependencies, formal controls and informal controls). For example, "IS solution consisted of modules, which had fewer interdependencies with other modules" was identified as one characteristic of 'low' level of interdependencies between modules. Initial characteristics of concepts were informed by the prior literature. Following the coding guideline, the interview data was assigned in to nodes in NVivo. During the coding process, coding guideline was enhanced by including new characteristics identified. Those characteristics were included in the coding guideline after careful consideration of definitions of each concept. The data analysis was conducted in two phases: 1) within-case analysis, and 2) cross-case analysis. Within-case analysis provided the ability to identify unique patterns in each case. This was followed by cross-case analysis to investigate the similarities and differences between the cases.

Results and Discussion

A first look at the case data revealed that the ISD projects included different levels of interdependencies (i.e. coupling) between modules. The focus of this research was to identify the interdependencies between modules, hence interdependencies within modules (i.e. cohesion) was not examined in this research. The interview questions related to modularization, formal control and informal-clan control, were designed to cover the entire project, rather than focusing on a specific module. Thus, the interview data captured the individuals' opinions about the entire project.

It was realized that the interdependencies between the modules in some of the case projects were not properly identified. For example, during Project A, some initial requirements of the project were removed as those requirements could not be implemented without implementing interdependent requirement. When the interdependencies between requirements were not properly identified, it can cause issues in project implementation. Respondent 01 from Project A explained: "*This requirement cannot be implemented without that* [requirement][4]*, because it clashed with other requirement… So, a big requirement was removed.*" When writing BRSs, business analysts should identify the

[3] Informant 02 was project manager for both Project A and Project G.

[4] The non-italicized portion within brackets (here and elsewhere) has been added to the transcript quotations to contextualize the quotation and make it clear for the readers.

interdependencies between different modules. Interdependencies were not properly identified by the business analysts in Project D. According to Respondent 10 from Project D: "*If we* [business analysts] *are writing the specification* [BRS], *we have to analyse the impact areas. Those areas were not clearly analysed.*" In project C, when team members were required to integrate a new module to the existing modules, they were required to consider about interfaces that connect the new module with the existing modules. This indicates that project C consisted of 'low' interdependencies. Respondent 09 from project C stated; "*We need to get the interface done and then test the requirement and the functionality.*" Software engineers of project E were assigned to different modules. Therefore, it was required for the software engineers to take the responsibility of the assigned software modules. Respondent 13 from project E stated; "*When a developer* [software engineer] *takes the responsibility of one component* [module] *he has the responsibility of changing the product document.*"

Table 1 presents the estimation of the level of interdependencies in each project. The level of interdependencies were identified based on the interview data and by analyzing BRSs. The level of interdependencies was identified by following characteristics; 1) modules, which have an impact on other modules, 2) modules, which require input from other modules, 3) interfaces between modules, 4) functionalities of the information system solution are sub-divided to modules, 5) team members are assigned to different modules, 6) team members' tasks are sub-divided, 7) team members' tasks, which require input from other team members' tasks and 8) team members are provided with isolated tasks, where the team members are required to complete the tasks independently.

Table 1: The level of interdependencies between modules

Project	**Level of interdependencies between modules**
A	High
B	High
C	Low
D	High
E	Low
F	Low
G	Low
H	Low

Propositions Testing

P1: When the level of interdependencies between the modules is low, it is likely that the projects will execute formal controls

In order to identify whether projects execute formal controls when there is low level of interdependencies between modules, it is required to recognize the level of formal control in each project. The level of formal controls were identified by following characteristics; 1) expected outcomes and behaviours are written in the BRSs, 2) updates to outcomes and behaviours are specified in BRSs, 3) similarities between the information system solution and the BRSs, 4) project deadlines, 5) project delays, 7) project meetings and 8) project plans. While for some of the case projects the level of formal control was high, for some of the projects it was low. For example, Project E included high formal controls. After the commencement of Project E, some decisions related to ISD procedures were taken at the meetings. The software engineering team did not proceed with the required ISD until the business analysts had updated the written specifications. This indicated high formal control in Project E. Respondent 15, a senior business analyst in Project E, stated: "*Most of the time, they* [software engineers] *are waiting for us* [business analysts] *to update the specifications* [BRSs] *and then proceed. Because, they* [software engineers] *will not proceed without us* [business analysts] *updating the document.*" The Project F team members were required to follow the BRSs strictly. Furthermore, business analysts did not aggregate the team members' ideas or feedback to the BRSs. This highlights that Project F had a high level of formal controls. Respondent 16 from Project F stated: "*We follow the BRS in a one to one basis.*" When the implementation of some modules was not feasible, the team members had to identify alternative methods to accommodate the particular module. Due to the changes, team members had to update the BRSs. This indicated that Project A was governed with less formal controls. Respondent 01 from Project A stated: "*Sometimes during developer discussions, they say this* [module] *is not feasible. During that discussion itself, you have to come up with* [an] *alternative to cater* *the functionality* [module]. *So after that what we* [business analysts] *did was, we just updated the BRSs and sent another version."* The BRSs of project C lacked information about the module interdependencies and the client business requirements. This indicated lack of formal controls in project C. Respondent 07 from project C stated; "*Most of the time, the spec* [BRS] *carry out only high-level requirements. If you take* [BRSs in] *another project very, very detailed than* [our project]*."* Since the BRSs did not include sufficient information on the module interdependencies, the team members of Project D were able to think in different ways. This indicated a less of formal controls in Project D. Respondent 10 from Project D stated: "*Documents* [BRSs] *do not provide some example or do not specify the areas* [modules]*. Then, we* [can] *think in several ways. So, that is the main thing."*

Cross-Case Analysis Summary

Since the projects C, E, F, G and H had low levels of interdependencies between modules, it was expected that these projects will have high levels of formal controls. As expected, the level of formal controls was high in project E, F and G. Thus, projects E, F and G supported the proposition 1. Although projects C and H consisted of low levels of interdependencies between modules, there were other factors such as project practices and volatile client requirements that caused less formal control in those projects. The BRS of project C included only the high level information about the module interdependencies and the project requirements. Moreover, the business analysts updated the BRSs including the suggestions from the software engineers. This indicates that project C followed a flexible project practice, where the software engineers were given the ability to provide suggestions to enhance the BRSs. In project H, the business analysts requested software code changes due to the volatile client requirements. Thus, project C and H challenged the proposition 1.

Since the project A, B and D had high levels of interdependencies between modules, it was expected that these projects will have low levels of formal control. As expected, the levels of formal control were low in projects A, B and D. While there were frequent updates to BRSs of project A, the software designs were updated in project B. Since the business analysts were unable to identify the module interdependencies accurately, the BRS of project D lacked information about the module interdependencies. Thus, projects A, B and D supported the proposition 1.

In summary, while the proposition 1 was supported in projects A, B, D, E, F and G, the proposition was challenged in projects C and H.

P2: When the level of interdependencies between the modules is low, it is unlikely that the projects will execute informal-clan controls

In order to identify whether the projects execute informal-clan controls when there are low levels of interdependencies between modules, it is required to recognize the level of informal-clan control within each project. The level of informal-clan control was identified using following characteristics; 1) the level of team spirit, 2) the level of similar values and beliefs, 3) the level of common goals or 4) the level of interactions and collaborations between the team members. When the BRSs included some modules that were not feasible, the team members had to identify alternative methods for achieving the project outcomes by conducting discussions with the team. According to Respondent 01 from Project A: "*Sometimes during developer discussions, they say this* [module] *is not feasible. During that discussion itself, you have to come up with* [an] *alternative to cater the functionality* [module]*."* Although there were collaborations between the team members in required situations, the team spirit of the project team members was low. Respondent 01 stated; "[In this project] *the team spirit was bit like less or minimum."* Although a software project is modularized, it is typical for a project to experience unplanned situations, so that collaboration between team members is required. Due to the high team spirit, the team members of Project B were able to handle the unplanned situations without much difficulty. Respondent 06 from Project B stated: "*I think this is a good team. We are working together without any issues. Developers* [software engineers]*, supporters, and QA* [quality assurance] *engineers are working together."* In situations where a particular team member was unable to complete the tasks before the deadline, other team members provided the support for completing the tasks. Respondent 22 from project H stated; "*In* [our project] *we have a very good team. The team is much bigger now. ... Normally everybody ... knew about each other strengths and weaknesses as well. Therefore, we don't* *have much problem to work late night or weekends or problems don't occur. Somebody fails to do* [a] *certain delivery or something; we* [were able to] *manage* [the situation using] *the available team members."*

Cross-Case Analysis Summary

Since the projects C, E, F, G and H consisted of low levels of interdependencies between modules, it was expected that these projects will have low levels of informal-clan control. Although project G consisted of low level of informal-clan control, project C, E, F and H consisted of high level of informal-clan control. Since the technical leads involved in the modularization process, it was not required for the software engineers of project G to have much collaborations with the business analysts. Thus, project G supported the proposition 2. In contrast, the team members of projects C, E, F and H were required to collaborate with the business analysts in order to get clarifications about the client requirements and the module interdependencies. For example, in project F, although the BRSs consisted of sufficient information for the ISD, team member collaborations were required to increase the understanding about the business requirements. In project H, in situations where a particular team member was unable to complete the tasks on time, other team members provided support. Moreover, in project H although the team members were assigned to different modules, sometimes they were required to collaborate with each other. Since the team members were located close by, the collaborations were much easier. This indicated that, although low interdependencies between modules minimize the need for clan control, there can be situations, where the team member collaborations are required. Thus, projects C, E, F and H challenged the proposition 2.

Projects A, B and D included high level of interdependencies between modules hence it was expected that the projects will include high levels of informal-clan control. For example, when the team members' tasks are interdependent, it will be required for the team members to collaborate with each other. Since projects B and D consisted of high informal-clan control, proposition 2 was supported in project B and D. In contrast, project A had low level of informal-clan control as a result of issues such as time pressure. Due to the time pressure, team members were trying to pass the responsibility of tasks to other team members. Therefore, proposition 2 was challenged in project A.

In summary, while the projects B, D and G supported the proposition 2, the projects A, C, E, F and H challenged the proposition.

Identification of Fluctuations in BRSs

Results of the interview data analysis indicated that the occurrence of an error in identification of module interdependencies at the initial stages of an ISD project created fluctuations in BRSs, which in turn, created issues for project management. For example, due to an error in identification of module interdependencies, some requirements of Project A were removed later in the ISD lifecycle. Respondent 01 from Project A stated: "*This requirement cannot be implemented without that [*requirement], *because it is clashed with other requirement…So, a big requirement was* ***removed."*** Since the BRSs were updated after removing the requirements, the software engineers had to amend the software code accordingly. Respondent 02 from Project A stated: "*They* [software engineers] *have to change certain things, because the document* [BRS] *is changing, it is changing continuously. It is very frequently changing. So, the developer* [software engineers] ***can't always accommodate the changes kind of****…* ***It is not easy****. It is not the proper practice."* In the initial stages of the projects, business analysts are required to conduct the modularization appropriately by identifying the interdependencies between the software requirements. The BRSs should include sufficient information about the module interdependencies. In situations, where the business analysts identify the module interdependencies later in the ISD lifecycle, it may be required to update the existing requirements or remove the requirements from modules. Subsequently, the software engineers have to update the software code. As software engineers highlighted, it is difficult to update the software code, later in the ISD lifecycle. Respondent 13 from project E stated: "*If they* [software engineers] *start the coding, it is very hard to include something that they* [business analysts] *didn't tell us* [software engineers] *in the early stages, and then, it* ***won't be possible****."* Since the interdependencies between modules were not properly identified, team members of project D had difficulties in testing. The team members were unable to conduct the software testing in a structured manner. Respondent 10 from project D stated: "*If we are* *writing the specification* [BRS] *we have to analyse the impact areas. Those areas were not clearly analysed. It is a problem in testing level because when we are going to test, we are jumping to the work right. So, then we are facing* ***some more issues****.”*

Analysis of BRSs of project A indicated that an error in identification of module interdependencies creates fluctuations in BRS towards the later stages of the project lifecycle. This was visible in fund processing and trade processing BRSs. Fund processing BRS consisted of information related to processing funds as per executed trades (e.g. processed transactions through ledger accounts and provided current cash positions to clients and brokering firms). Trade processing BRS explained the entire life cycle of trades including trade entry, trade amendments, trade splits, trade confirmations and trade rejections. As specified in the revision history of fund processing BRS, on 5th December 2011, the accounting structure of project A had to be removed as the accounting structure could not be implemented without implementing another interdependent module. As a result of removing the accounting structure, the business analysts had to add or delete 116 spec points[5], amend 73 points and clarify 29 points of fund processing BRS. Removing accounting structure created the need of more updates in fund processing BRS towards the end of the life cycle. For example, on 13th August 2012, fund processing BRS was updated by: 1) adding or deleting 85 spec points; 2) amending 33 spec points; and 3) clarifying 31 spec points. See table 2 for the number of updates in fund processing BRS.

Table 2: Number of updates in fund processing BRS

Fund processing			
Date	Number of spec point updates based on the criticality level		
	Added / Deleted*	Amended**	Clarified***
07/06/2011	6	8	0
14/06/2011	1	6	2
22/06/2011	19	20	8
05/12/2011	116	73	29
26/02/2012	7	3	1
03/03/2012	21	15	0
31/05/2012	26	14	0
13/08/2012	85	33	31
27/02/2013	20	6	0

Added / Deleted* - a spec point was added or deleted from BRS; Amended** - a spec point was amended; Clarified*** - a spec point was clarified including clarification information.

Since trade processing BRS had interdependencies with fund processing BRS, it was required to update the trade processing BRS as well. As a result of fund processing BRS updates, 92 trade processing spec points were added, 35 amended and 26 clarified on 02nd February 2012. Table 3 provides the number of updates in trade processing BRS.

Table 3: Number of updates in trade processing BRS

Trade processing			
Date	Number of spec point updates based on the criticality level		
	Added / Deleted	Amended	Clarified
20/09/2011	38	10	0
31/10/2011	19	5	3
02/02/2012	92	35	26
02/03/2012	2	14	0
25/06/2012	13	15	0
25/07/2012	37	37	0

Further exploration of the BRSs showed that the BRSs of case projects were sharing requirements. For example, when the fund processing module was relevant to the trade processing module and user management module, the requirements related to the fund processing module were included in both trade processing BRS and user management BRS. This parallels the “component sharing modularity” suggested by Pine (1993). According to Pine (1993), in component sharing modularity, the same

[5] Spec point- a description written under a specific number in BRS (e.g. section 2.2.1 – update trade postings).

component is shared across multiple modules. Similarly, in the ISD projects investigated in this study, the same requirements were shared across multiple BRSs (see figure 2). Therefore, a change in requirements in one BRS created several changes to other BRSs.

On the basis of the study's findings, it is suggested that sectional modularity can be used to minimize the fluctuations in BRSs. According to Pine (1993, p. 208), sectional modularity provides "the greatest degree of variety and customization... [it] allows the configuration of any number of different types of components in arbitrary ways-as long as each component is connected to another at standard interfaces." In the ISD project context, it is recommended that BRSs has to be documented following the sectional modularity, where each module includes a separate BRS. When a particular module has to be shared between other modules, the BRSs of other modules should include only the inputs and outputs of the particular module. All the information related to a particular module should only be included in the BRS related to that particular module (see figure 2). When BRSs are documented following the sectional modularity, a change in one BRS has less impact on the other BRSs. Thus, the fluctuations in BRSs can be minimized, which in turn minimize project management issues.

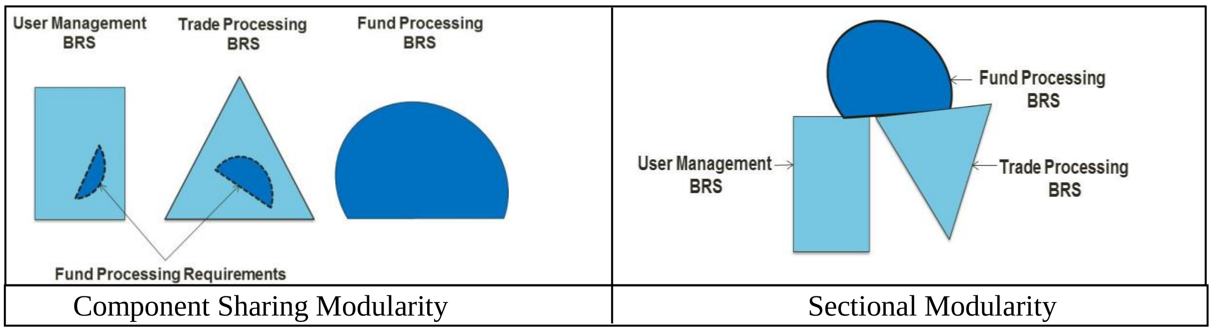

Figure 2: Component sharing modularity and sectional modularity

Implications for Research

Providing a theoretical extension to control theory, this study highlights the importance of considering interdependencies when managing ISD projects. Kirsch (2004) points out that the different stages of ISD projects include task interdependencies, leading to changes in the choices for control mechanisms. According to the study's findings, when the projects consisted of low level of interdependencies between modules, project managers tend to use formal controls to govern the team members. Although some projects included low level of interdependencies between modules, there can be other factors such as project practices and volatile client requirements that may minimize the level of formal control in projects. Research findings indicate that utilizing sectional modularity instead of component sharing modularity can minimize project management issues. This extends previous findings on relationships between task interdependencies and project control (Kirsch 2004; Ouchi and Maguire 1975). Our findings add further substance to the claims presented in Tiwana (2008), who explained how minimizing interdependencies between outsourced systems and other software applications is a substitute for controls. While such studies made a substantial contribution to our knowledge, a theoretical explanation for the control mechanisms suitable for managing modularized ISD projects remains scant. As such, we extend past research (Choudhury and Sabherwal 2003; Tiwana 2008) by highlighting the importance of selecting control mechanisms considering interdependencies between modules.

Kirsch (1996) recommends organizations utilize clan control in order to increase individuals' commitment to the team. Although the control literature discusses clan control, it does not discuss the management of multiple clans within a single ISD project. This study's results indicated that ISD projects consist of several teams with different team goals and objectives, thereby creating multiple clans within a single project. Moreover, these teams are assigned to different modules. Therefore, when the projects include module interdependencies, project managers should ensure that clan control is appropriately implemented within teams (e.g. within the business analysts team and within the software engineering team) and between teams (e.g. between business analysts team and the software engineering team) assigned to different modules. As expected, the use of informal-clan control in projects was minimized due to low interdependencies between modules. Even though some projects consisted of low interdependencies between modules, clan control was required in order to gain the required level of information about the client requirements.

Though the study makes several theoretical contributions, it has a few limitations that need to be acknowledged. First, although procedures for internal and external validity were followed, the subjectivity of data collection and analysis may be an issue. Second, the study did not validate the changes in BRS fluctuations based on the type of modularity. Therefore, future research can examine the BRS fluctuations according to the type of modularity. Third, the study's sample consisted of eight projects selected from a single ISD organization which might preclude generalizability. Thus, it would be beneficial for other studies to use samples from multiple organizations in order to enrich the obtained insights about the effective control mechanisms for modularized ISD projects. Fourth, a longitudinal study on modularized ISD project control could yield further insights about the phenomenon observed. For example, changes in the BRS fluctuations would be better explained through a longitudinal study. Fifth, future researchers could explore modularization in the context of distributed ISD project teams. Increasingly distributed nature of ISD projects (McCarthy et al. 2018) demands for such research. Finally, consideration of the ISD method (i.e. agile or waterfall) adopted (Russo et al. 2013) in each case could offer further insights about the phenomenon.

Implications for Practice

In addition to implications for research, this study has potential to influence practice of ISD project management. Results from the study highlight that when projects consisted of low level of interdependencies between modules, project managers tend to use formal control to govern the team members. Although projects include low level of interdependencies between modules, informal-clan control is required to ensure that the team members understand the client requirements accurately. Therefore, project managers should have a proper understanding about control mechanisms for managing modularized ISD projects. As a result of assigning team members to different modules, projects consisted of multiple clans. However, there can be interdependencies between modules. Thus, project managers should ensure all teams of project are working collaboratively as a single team. This research revealed that when projects include component sharing modularity, it can create project management issues towards the end of the project lifecycle. Thus, project managers should pay more attention about the project controls in the final stages of the projects. Rather than utilizing component sharing modularity, use of sectional modularity enables efficient control in projects. Therefore, when documenting the BRSs, business analysts must ensure that the BRSs are documented using the sectional modularity.

References

Adamo-Villani, N., Antsaklis, P. J., Aragon, C. R., Bagheri, N., Baiden, G., Balasubramanian, P., and Zak, S. H. 2009. "Handbook of Automation ", S.Y. Nof (ed.). Springer.

Al-Otaiby, T. N., AlSherif, M., and Bond, W. P. 2005. "Toward Software Requirements Modularization Using Hierarchical Clustering Techniques," *Southeast Regional Conference,* Kennesaw, Georgia: ACM, pp. 223-228.

Allen, E. B., and Khoshgoftaar, T. M. 1999. "Measuring Coupling and Cohesion: An Information-Theory Approach," *Sixth International Software Metrics Symposium*: IEEE, pp. 119-127.

Benbasat, I., Goldstein, D. K., and Mead, M. 1987. "The Case Research Strategy in Studies of Information Systems," *MIS Quarterly* (11:3), pp. 369-386.

Cataldo, M., Herbsleb, J. D., and Carley, K. M. 2008. "Socio-Technical Congruence: A Framework for Assessing the Impact of Technical and Work Dependencies on Software Development Productivity," in: *ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*. Kaiserslautern, Germany: ACM, pp. 2-11.

Cataldo, M., and Nambiar, S. 2012. "The Impact of Geographic Distribution and the Nature of Technical Coupling on the Quality of Global Software Development Projects," *Journal of Software: Evolution and Process* (24:2), pp. 153-168.

Chou, S.-W., and He, M.-Y. 2011. "The Factors That Affect the Performance of Open Source Software Development - the Perspective of Social Capital and Expertise Integration," *Information Systems Journal* (21:2), pp. 195-219.

Choudhury, V., and Sabherwal, R. 2003. "Portfolios of Control in Outsourced Software Development Projects," *Information Systems Research* (14:3), pp. 291-314.

Clark, K. B., and Baldwin, C. Y. 2000. *Design Rules : The Power of Modularity*. Cambridge, Massachusetts: MIT Press.

Dedrick, J., Carmel, E., and Kraemer, K. L. 2011. "A Dynamic Model of Offshore Software Development," *Journal of Information Technology* (26:1), pp. 1-15.

Dibbern, J., Winkler, J., and Heinzl, A. 2008. "Explaining Variations in Client Extra Costs between Software Projects Offshored to India," *MIS Quarterly* (32:2), pp. 333-366.

Dingsøyr, T., Nerur, S., Balijepally, V., and Moe, N. B. 2012. "A Decade of Agile Methodologies: Towards Explaining Agile Software Development," *Journal of Systems and Software* (85:6), pp. 1213-1221.

Eisenhardt, K. M. 1985. "Control: Organizational and Economic Approaches," *Management Science* (31:2), pp. 134-149.

Gershenson, J., Prasad, G., and Zhang, Y. 2003. "Product Modularity: Definitions and Benefits," *Journal of Engineering Design* (14:3), pp. 295-313.

Goulão, M. 2001. "Coupling and Cohesion as Modularization Drivers: Are We Being over-Persuaded?," *Fifth European Conference on Software Maintenance and Reengineering*, Lisbon, Portugal: IEEE, pp. 47-57.

Haag, S., Raja, M. K., and Schkade, L. L. 1996. "Quality Function Deployment Usage in Software Development," *Communications of the ACM* (39:1), pp. 41-49.

Hoegl, M., Parboteeah, P. K., and Gemuenden, H. G. 2003. "When Teamwork Really Matters: Task Innovativeness as a Moderator of the Teamwork–Performance Relationship in Software Development Projects," *Journal of Engineering and Technology Management* (20:4), pp. 281-302.

Jaworski, B. J. 1988. "Toward a Theory of Marketing Control: Environmental Context, Control Types, and Consequences," *Journal of Marketing* (52:3), pp. 23-39.

Kirsch, L. J. 1996. "The Management of Complex Tasks in Organizations: Controlling the Systems Development Process," *Organization Science* (7:1), pp. 1-21.

Kirsch, L. J. 2004. "Deploying Common Systems Globally: The Dynamics of Control," *Information Systems Research* (15:4), pp. 374-395.

Klein, H. K., and Myers, M. D. 1999. "A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems," *MIS Quarterly* (23:1), pp. 67-93.

Kraut, R. E., and Streeter, L. A. 1995. "Coordination in Software Development," *Communications of the ACM* (38:3), pp. 69-81.

Kumar, N., Stern, L. W., and Anderson, J. C. 1993. "Conducting Interorganizational Research Using Key Informants," *Academy of Management Journal* (36:6), pp. 1633-1651.

Lee, A. S. 1989. "A Scientific Methodology for Mis Case Studies," *MIS Quarterly* (13:1), pp. 33-50.

Manz, C. C., and Angle, H. 1986. "Can Group Self-Management Mean a Loss of Personal Control: Triangulating a Paradox," *Group & Organization Studies* (11:4), pp. 309-339.

McCarthy, S., O'Raghallaigh, P., Fitzgerald, C., and Adam, F. 2018. "Theorising Antecedents of Cohesion and Conflict in Distributed Isd Project Teams," *Proceedings of the 39th International Conference on Information Systems*: Association for Information Systems (AIS).

Mikkola, J. H., and Skjøtt-Larsen, T. 2004. "Supply-Chain Integration: Implications for Mass Customization, Modularization and Postponement Strategies," *Production Planning & Control* (15:4), pp. 352-361.

Minichiello, V., Aroni, R., Timewell, E., and Alexander, L. 1995. *In-Depth Interviewing: Principles, Techniques, Analysis*, (2nd ed.). Melbourne, Australia: Pearson Education Australia.

Nuwangi, S. M. 2016. "The Impact of Modularisation on Information System Development Outsourcing Project Control," in: *Information Systems*. Queensland University of Technology, Australia p. 259.

Nuwangi, S. M., Sedera, D., and Srivastava, S. C. 2014. "Introducing System Controls for Control Theory," *Australasian Conference on Information Systems*, Auckland, New Zealand.

Nuwangi, S. M., Sedera, D., and Srivastava, S. C. 2018. "Multi-Layered Control Mechanisms in Software Development Outsourcing," *Pacific Asia Conference on Information Systems*, Yokohama, Japan: Association for Information Systems AIS Electronic Library (AISeL), pp. 1-9.

Ouchi, W. G. 1978. "The Transmission of Control through Organizational Hierarchy," *Academy of Management Journal* (21:2), pp. 173-192.

Ouchi, W. G. 1980. "Markets, Bureaucracies, and Clans," *Administrative Science Quarterly* (25:1), pp. 129-141.

Ouchi, W. G., and Maguire, M. A. 1975. "Organizational Control: Two Functions," *Administrative Science Quarterly* (20:4), pp. 559-569.

Patton, M. Q. 2002. *Qualitative Research and Evaluation Methods*. Thousand Oaks, Calif: Sage.

Pflügler, C., Wiesche, M., and Krcmar, H. 2018. "Subgroups in Agile and Traditional IT Project Teams," *51st Hawaii International Conference on System Sciences*.

Pine, B. J. 1993. *Mass Customization the New Frontier in Business Competition*. Boston, Massachusetts Harvard Business School Press

Ravishankar, M. N., and Pan, S. L. 2013. "Examining the Influence of Modularity and Knowledge Management (KM) on Dynamic Capabilities: Insights from a Call Center," *International Journal of Information Management* (33:1), pp. 147-159.

Russo, N. L., Fitzgerald, G., and Shams, S. 2013. "Exploring Adoption and Use of Agile Methods: A Comparative Case Study,").

Rustagi, S., King, W. R., and Kirsch, L. J. 2008. "Predictors of Formal Control Usage in IT Outsourcing Partnerships," *Information Systems Research* (19:2), pp. 126-143.

Sanchez, R. 1995. "Strategic Flexibility in Product Competition," *Strategic Management Journal* (16), pp. 135-159.

Sanchez, R., and Mahoney, J. T. 1996. "Modularity, Flexibility, and Knowledge Management in Product and Organization Design," *Strategic Management Journal* (17:Winter Special Issue), pp. 63-76.

Schmidt, R., Lyytinen, K., Keil, M., and Cule, P. 2001. "Identifying Software Project Risks: An International Delphi Study," *Journal of Management Information Systems* (17:4), pp. 5-36.

Sedera, D., Lokuge, S., Krcmar, H., Srivastava, S. C., and Ravishankar, M. N. 2014. "The Future of Outsourcing in the Asia-Pacific Region: Implications for Research and Practice—Panel Report from PACIS 2014," *Communications of the Association for Information Systems* (35:1), pp. 317-331.

Senarath, S. 2016. "Not So 'Bankruptcy-Remote': An Insight into Sri Lankan Securitization Practices in a Post_Gfc Context," in: *Multidisciplinary Academic Conference on Management, Marketing and Economics*. pp. 53-60.

Senarath, S. 2017. "The Dodd-Frank Act Doesn't Solve the Principal-Agent Problem in Asset Securitisation." LSE Research Online: LSE Business Review.

Senarath, S., and Copp, R. 2015. "Credit Default Swaps and the Global Financial Crisis: Reframing Credit Default Swaps as Quasi-Insurance," *Global Economy and Finance Journal* (8:1), pp. 135-149.

Sosa, M. E., Eppinger, S. D., and Rowles, C. M. 2004. "The Misalignment of Product Architecture and Organizational Structure in Complex Product Development," *Management Science* (50:12), pp. 1674-1689.

Tiwana, A. 2008. "Does Technological Modularity Substitute for Control? A Study of Alliance Performance in Software Outsourcing," *Strategic Management Journal* (29:7), pp. 769-780.

Yin, R. K. 2003. *Case Study Research: Design and Methods*, (3rd ed.). Thousand Oaks, CA: Sage Publications.

Zimmermann, A., and Ravishankar, M. 2014. "Knowledge Transfer in IT Offshoring Relationships: The Roles of Social Capital, Efficacy and Outcome Expectations," *Information Systems Journal* (24:2), pp. 167-202.

Appendices

Appendix A – Project Descriptions

Project Name	Description of the Project
A	This project focused on developing a post-trade application, which provides clearing and settlement for the trades after execution. Functionalities include trade processing, user management, general accounting and journal entries. Moreover, the software application consists of complex trade processing methods which are highly integrated with clearing and settlement procedures. The client is situated in the Asian region.
B	This is a real-time clearing system that manages post-trade activities. This project consists of three major functionalities: 1) clearing (the process of matching, recording and transaction processing); 2) settlement (trade settlement); and 3) risk management. The client company is situated in Europe.
C	This project addresses a wide range of business needs including connectivity and order management. It can handle a variety of firms' trading business; covering front office, middle and back office functionality. Key characteristics of the information system solution include: connectivity hub and post-trade risk management. The project consists of around 40 clients all over the world including clients from North America and Asia.
D	This project involves developing a tested, real-time and transparent trading platform. Furthermore, it facilitates robust assaying and warehousing facilities to execute the trades. The key features of software solution include: hedging; and clearing and settlement. The client is situated in the Asian region.
E	This project involves the development of tools to detect irregular trading behaviors. Key functionalities of the information system include: analysis of real-time / offline transaction data; provide alert and case-management. This project consists of several clients including clients from Asian and African regions.
F	This is a project that integrates several trading platforms. It provides smooth transition for trading systems. This project is focused on developing a platform for a client, who was formed in 1980s. This client provides capital market solutions for several instrument trading.
G	This project includes the development of a post-trade application. The post-trade application includes clearing and settlement of the executed trades. There are several users with different authorization levels; Registry owner is the main user. Client company is situated in the Asian region.

Project Name	Description of the Project
H	This project develops a highly critical application due to client-focused solutions and integrity of modules. The application can be adjusted to trade any product in any type of market. project consists of multiple clients from all over the world, including clients from Europe.

Appendix B – Interview Questions

1. Can you please describe Project X[6]?
2. What sort of issues do you encounter in your project?
3. Can you describe the documents and software systems that your team uses to transfer the client requirements?
4. To what level do you document the client requirements?
5. Are there any other documents and systems that your team uses as contracts between you and client?
6. Can you please describe the penalties, rewards and time allocations of your project?
7. Can you discuss the issues you face when you are controlling a project?
8. What knowledge is required for your team members to develop products that satisfy client requirements?
9. What knowledge do team members have about the contracts and requirement documents?
10. To what level do you follow the document during day-to-day activities?
11. How do you describe the behavior of your project team members? Are they flexible to provide more information than what is mentioned in the requirement documents?
12. Can you please describe the team spirit, shared values and beliefs of the team?
13. To what extent do you amend the requirement documents, time estimations and project templates according to the requests from the team members?
14. How does your team manage day-to-day operations and quick decisions?

Appendix C – Participant Information

Project	Participant ID	Designation	Years of experience in SD industry	Years of experience in the company
A	01	Senior Business Analyst	4	4
	02	Project Manager	15	2
	03	Specialist Software Engineer	8	8
B	04	Director Business Operations (Post-Trade)	11	10
	05	Senior Tech Lead	7	7
	06	Senior Business Analyst	4	4
C	07	Business Analyst	4	3
	08	Principal Software Engineer	5	4
	09	Junior Project Manager	4	3
D	10	Business Analyst	8	8
	11	Technical Lead	9	9
	12	Associate Project Manager	10	10
E	13	Senior Software Engineer	8	8
	14	Junior Project Manager	3	3
	15	Senior Business Analyst	3*	3*
F	16	Technical Lead	12	9
	17	Project Manager	9	9
	18	Senior Business Analyst	6	4
G	19	Specialist Software Engineer	8	8
	20	Consultant	12	10
	02**	Project Manager	15	2
H	21	Project Manager	6*	5*
	22	Senior Software Engineer	3*	3*
	23	Senior Business Analyst	3*	3*

*Verified through LinkedIn
**Informant 2 was the project manager for both Project A and Project G.

[6] To maintain confidentiality, the names of the projects were disguised.