IPRPD
International Journal of Business & Management Studies
ISSN 2694-1430 (Print), 2694-1449 (Online)
Volume 05; Issue no 06: June, 2024
DOI: 10.56734/ijbms.v5n6a6

THE ROLE OF MODULAR ARCHITECTURE IN THE CONTEXT OF IS/IT PROJECT OUTSOURCING: AN EMPIRICAL ANALYSIS

Dr. Shahzada Benazeer[1], Prof. Dr. Jan Verelst[2], Dr. Philip Huysmans[3]

[1]*University of Antwerp, Belgium*

[2]*Head of the Department, Department of Management Information Systems, University of Antwerp, Belgium*

[3]*University of Antwerp, Belgium*

Abstract

The outsourcing of information systems and information technology (IS/IT) has become a prevalent practice in both developed and emerging economies, with the aim of achieving economies of scale and competitive advantages. In general, an IS/IT outsourcing deal concerns an agreement or contract (service level agreement or SLA) between parties in which one party (the vendor organization) agrees to deliver certain services to another party (the customer organization). The outsourcing literature is persistently reporting high failure rates in IS/IT project outsourcing and suggests that the IS/IT project outsourcing process is a complex maneuver. However, it is likely that due to inappropriate strategies and short-sightedness, at least a third of IS/IT outsourcing projects fail to deliver intended targets. For decades, in numerous fields, the issue of complexity has been addressed by applying the concept of modularity. Modular architecture is a key aspect of the concept of modularity. An in-house team of skilled individuals possessing expertise in architectural and systems integration knowledge may help to achieve competitive advantage by effectively managing complexity. In addition to achieving economies of scale, the customer organization may also achieve agility and flexibility by combining the advantages of modularity and outsourcing. This study examines the feasibility of applying the concept of modularity in the context of IS/IT project outsourcing with the aim of further expanding its scope. The findings of empirical analyses of four cases suggest that numerous aspects of the concept of modularity are relevant in the context of IS/IT project outsourcing. The 'modular architecture' has emerged as one of the most relevant and was identified in three out of the four analyzed cases. It implies that the 'modular architecture' aspect ought to receive greater attention when designing or planning a new IS/IT outsourcing project. The authors argue by stating that the present study has provided support for the point of view that engineering concepts such as modularity can be applied with substantial relevance to research domains which are predominantly considered from an IT management-perspective. Perhaps the use of the same concept, i.e., modularity, at both the technical and management level will provide a common vocabulary and may be even more to achieve better alignment between both levels and contribute to realizing more successful IS/IT outsourcing projects in the future. However, at the very least, a common vocabulary should help in collaboration between researchers at both levels to advance the research and practice in IS/IT project outsourcing, to which this is an invitation. It would be a contribution to the new and emerging research area focusing on applying engineering concepts to the design of enterprises, coined Enterprise Engineering. In addition to theoretical contributions, this study has implications at a practical level by providing an alternative lens for the CIO's to analyze their future IS/IT projects.

Keywords

Module, Systems-Integration, Complexity, Knowledge, Fine-Grained, Enterprise Engineering

Introduction

The digitalization and automation of systems utilized by businesses, industries and services in a global village is a phenomenon that implies that certain processes should be managed by external partners who are technically more efficient and capable of providing economies of scale. This straightforward logic prompts the decision-making units of an organization to contemplate the possibility of outsourcing. Nonetheless, it is considerably less challenging to contemplate than to execute this strenuous procedure. Outsourcing of IS/IT projects became a common practice among contemporary organizations in developed and in emerging economies. Literature suggests over 94% of '*Fortune 500*' companies are outsourcing at least one major business function (Modarress, Ansari, & Thies, 2014). Despite the widespread use and extensive expertise of Chief Information Officers in IS/IT project outsourcing, the failure of such projects is a prevalent occurrence. The literature suggests that at least one in three projects were considered a failure and that, in addition to this, many projects were delayed, ran over budget, and were not able to meet their pre-defined targets (Delens, Peters, Verhoef, & Van Vlijmen, 2016; Jabangwe, Smite, & Hesbo, 2016; Schmidt, Zoller, & Rosenkranz, 2016; Wojewoda & Hastie, 2015). A pertinent question deals with how IS/IT project outsourcing failure may be addressed. In order to find an answer or answers to this challenging question, numerous scholars proposed different approaches. For instance, Peterson and Carco (1998) suggested to streamline operations and 'fix the problem' before outsourcing IS/IT services. Various suggestions were introduced: the interested reader is referred to (1) Lambert, Emmelhainz, and Gardner (1999) who introduced their '*Partnership Model*'; (2) Greaver (1999) who formulated '*seven steps to successful outsourcing'*; (3) Logan (2000) who proposed two solutions in order to avoid failure in IS/IT project outsourcing. She suggests firstly, diagnosing the relationship from both sides of the contract and secondly, engaging agency theory to help design the types of contracts and relationships necessary to provide and support an environment of trust; (4) Lee (2001) who suggested knowledge sharing; (5) Rottman (2008) who elaborates on the importance of 'knowledge transfer'; (6) Harris, Herron and Iwanicki (2008) who stressed the importance of a high quality 'service level agreement' (SLA); (7) Karimi-Alaghehband and Rivard (2012) who proposed a model of IS/IT outsourcing success grounded in dynamic capabilities perspective; (8) Ishizaka and Blakiston, (2012) who proposed the *"18 C's model"* for a successful long-term outsourcing arrangement; and (9) Zheng and Abbott (2013) who argued that reconfiguration of organizational resources is vital to be successful in outsourcing. Despite the implementation of such measures, empirical evidence continues to demonstrate the high rate of failure in IS/IT project outsourcing. It appears that these remedies, if employed, proved to be partially efficacious at best. A properly designed IS/IT outsourcing project can bring tremendous advantages to the customer organization. Failure to do so will result in the entire advantage being lost to the service provider organization due to '*vendor lock-in*', resulting in irreversible damages to the customer organization. The advantages for the customer organizations include not only economies of scale but also the ability to free itself from stretched resources, gain agility and flexibility if the systems can be transformed into a '*black-box*', and provide opportunities to focus on core competencies. When a system is commoditized due to a proficient modular architecture, only a limited number of skilled personnel are required to operate or maintain it. By outsourcing noncore activities, customer organizations intend to disregard those activities and anticipate that the noncore activities will function as a '*black box*' or in a *'plug and play'* mode with minimal intervention. Due to the volatile business environment, contemporary organizations are under immense pressure to achieve greater agility and flexibility in order to adapt to the ever-changing business environment. In this emerging volatile and ever-changing situation, contemporary organizations are splitting up their IS/IT systems, outsourcing agreements, and organizational structures into modules. When IS/IT systems are split into many modules, it offers greater agility and flexibility to customer organizations to decide which module/s will remain in-house and which module/s can be outsourced. Moreover, the customer organizations get the options to decide to outsource all the modules to a single vendor or to multiple vendors. Hence, the modular structure of IS/IT systems offers a range of advantages including the ability to outsource IS/IT services to multiple vendors at a competitive price, and, in the event of necessity, the ability to substitute vendors or even back-source (reversibility) the IS/IT services. This is attributed to the notion that in a truly modular system, modules can be utilized either as a *'black-box'* or in a '*plug-and- play'* mode (Sako, 2005). The concept of modularity has been used in a variety of disciplines including 'Information Systems', Management and Organization Sciences, Engineering, Product Design, Production Systems, 'Psychology', 'Biology', and 'Mathematics' (Schilling, 2000). One of the consequences of the versatile utility of the concept of modularity is that even within modularity literature; the concept is used in different ways and applied to different units of analysis (Campagnolo & Camuffo, 2010). Some of the salient aspects of the concept of modularity include 'interface or SLA', 'modular architecture', 'cohesion', 'modular operator', 'coupling', 'dependencies', 'separation of concern', 'design rules', 'standards', and 'encapsulation' or 'information hiding'.

Literature Review

The IS/IT project outsourcing is often dealing with the software and/or business application developments. A study analyzing 22,031 IS/IT contracts signed during a period of 20 years (1989 to 2009); found that highly modularized projects are more likely to be outsourced to multiple vendors (Bapna, Gupta, Ray, & Singh, 2013). The literature also suggests that systematic modular architectural design of an IS/IT system is a precondition for a successful IS/IT outsourcing project. Sako (2014, p. 8-9) states that

> "Outsourcing/offshoring requires a certain degree of disaggregation, standardization, and modularization of tasks before services can be delivered from one legal entity to another at a geographic distance. It is a consequence also because the application of these operational techniques may be triggered by, and therefore follow, the decision to outsource/offshore".

Colfer and Baldwin (2016) conducted a descriptive study covering varied industries applying the 'mirroring hypothesis' and their findings suggest that modular systems with a reduced level of interdependencies are more likely to be outsourced. Similarly, MacDuffie (2013) contend that in the automotive industry, the managers tend to outsource those activities for which they have gained enough skills through experiences about the interdependencies and about specifying the interfaces. Moreover, numerous scholars assert that modular architecture plays a favorable role in the success of IS/IT project outsourcing. For instance, Nagpal and Lyytinen (2010, p. 2 & 5) assert that

> "A modular architecture enables a greater number of outputs to be made available as IT services, using a limited number of IT inputs. This can affect outsourcing success. Therefore, modular system theory is a relevant theoretical lens in this context. The link between outsourcing success and modularity is drawn on the basis of increased efficiency and effectiveness of the outsourcing process, when modular architectures are used in the IT organization".

In fact, in order to succeed in the ever-evolving business environment, contemporary organizations require a certain level of architectural knowledge and system integration knowledge (figure 1). The decomposition of a monolithic system into modules and/or updating/upgrading a legacy system necessitates architectural knowledge. The literature further elaborates in support of this argument as follows: *"the approach in the outsourcing category – is to decompose the product into components and modules"* (Eppinger & Chitkara, 2009, p. 7). However, an organization has to deal with numerous internal (in-sourced) and external (outsourced) modules, it is imperative to possess expertise in system integration knowledge at this stage. Eppinger and Chitkara (2009, p. 9 & 10) state that *"clearly defined interfaces between modules facilitate their separate development and eventual integration into the product"*. System integration knowledge for an organization is a prerequisite in order to be successful in outsourcing maneuvers. Zirpoli and Becker (2011, p.23) argue that *"Interface standards and modularity, of course, facilitate outsourcing and thereby sharpen requirements for integration"*. The subsequent section provides a brief overview of the research methodology employed and at the end formulates the research questions.

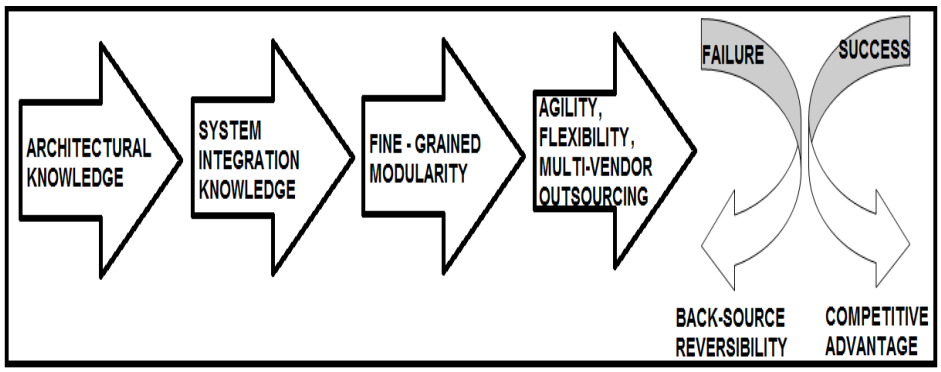

Figure 1. The importance of architectural and systems integration knowledge is paramount

Methodology and Research Questions

This investigation seeks to gain a deeper understanding of how the failures in IS/IT project outsourcing may be addressed by using the lens of modularity, which is rarely considered in academic discourse. Therefore, an interview-based descriptive, qualitative, and multiple case study research approach has been adopted as it seems to be one of the most appropriate methodological approaches for obtaining answers to the research questions. The repetition of an experiment enhances the validity and reliability of research findings. Similarly, in case study research, investigating multiple cases enhances the validity and reliability of research findings (Benbasat, Goldstein, & Mead, 1987; Yin, 2014). Yin (2014) suggests that a descriptive case study research approach is the most appropriate for understanding contemporary issues, especially when the boundaries between the phenomenon and the context are not clearly defined. In order to directly observe and acquire firsthand knowledge of the systems architecture, three investigators made numerous onsite visits and conducted multiple sessions of exploratory interviews that were open-ended, semi-structured. In addition to the primary data gathered from the sites and from the interviews, certain data were also obtained from other sources, such as documentation, archival records, and media outlets. A comprehensive, exploratory investigation has been conducted to investigate the phenomena, that can be interpreted as modular structures in the context of IS/IT project outsourcing. This study seeks to determine the relevance and importance of aspects of modularity that may be associated with success or failure factors in the IS/IT project analyzed cases. Therefore, answering the following research questions may help in elucidating the research objective.

RQ1: *Which instances (examples, counter-examples) of the use of modularity in the context of IS/IT outsourcing can be identified?*

RQ2: *How can the relevance and/or importance of these instances (examples, counter examples) for IS/IT outsourcing project be assessed?*

Modular Architecture

A modular architecture can be described as *"a predefined set of prescriptive rules which all modules of the system need to adhere to"* (Huysmans et al., 2014, p. 4418). The modular architecture is a design principle of any system that decomposes a complex system into loosely coupled modules. These loosely coupled modules work together respecting the predefined set of prescriptive rules in order to achieve a common goal. A modular architecture identifies the modules of a system and defines the conditions of an interface. The defined conditions of an interface may impose the rules how modules can communicate amongst each other. The IS/IT architecture of an organization refers to the arrangement through which various software applications and subsystems are interlinked (Kruchten, Obbink, & Stafford, 2006). Ideally, the modular architecture of an IS/IT project should be designed before starting the project and should not be changed during the implementation of the project as this usually causes a large change impact. However, the 'Normalized Systems Theory' (NST) proposed by Mannaert et al. (2012 & 2016) is a distinctive evolvable modular architecture that suggests certain design principles to mitigate or control 'combinatorial effects' or 'ripple effects'. A combinatorial effect occurs when the effort to apply a certain change to a system is not only dependent on that change itself, but also on the size of the system. The organization involved with the IS/IT project should strictly follow the predefined guidelines of the modular architecture. During the development or implementation stage, if there is a necessity to change the architecture, it should be executed by the same person who has designed it or by a person who has an in-depth knowledge about this particular architecture. On this issue Baldwin and Clark (2000, p. 127-128) assert that *"architectural changes do not arise naturally from incremental analysis [….] The knowledge needed to make these judgments is gained from experience with prior designs, combined with a detailed understanding of the artifact's intended functions and of its physical logical structure"*. More recently in an empirical study, Sanchez and Shibata (2021) proposed a set of ten '*Modularity Design Rules*' (MDRs) which are essential in developing modular architectures and should be considered during three proposed development phases. They state that "*the essential upstream managerial interactions and subsequent development activities need to be carried out within a clear framework of specific rules for governing and guiding a firm's processes for developing modular architectures*" (Sanchez & Shibata, 2021, p. 2). Based on the concept of modularity, numerous 'IS/IT architecture' and 'Enterprise Architecture' design methodologies have been proposed in the literature that can help create a flawless modular architecture. For instance, certain renowned design methodologies in both academia and industry include '*Normalized Systems*' (NS), '*Design Structure Matrix*' (DSM), '*Modular Operators*', and '*Design & Engineering Methodology for Organizations*' (DEMO). Some of these methods are briefly discussed in the subsequent section.

Normalized Systems

Changing and adapting their software systems is crucial for organizations in order to remain agile and flexible. However, several indications exist that this is hard to realize in practice. For instance, Lehman's Law of Increasing Complexity states that *"….as an evolving program is continually changed, its complexity, reflecting deteriorating structure, increases unless work is done to maintain or reduce it"* (Lehman, 1980, p. 1068). This deteriorating structure implies that applying similar changes to a system becomes more difficult over time. Normalized Systems (NS) tries to tackle this phenomenon by demanding BIBO (bounded input, bounded output) stability as defined in systems theory to software systems (Mannaert et al., 2012 & 2016). This means that the impact of a bounded set of changes to a software system should only depend on the type of the changes, not on the size of the system to which they are applied. Conversely, changes which are dependent on the size of the system are coined combinatorial effects (Mannaert et al., 2012 & 2016). In an agile environment, where we can assume that software systems are ever-growing and changing, such combinatorial effects become eventually prohibitive as their effort (and therefore, cost) may become larger than the cost for creating an entirely new software system. A system which is free of combinatorial effects is called a Normalized System (Mannaert et al., 2012 & 2016). NS proposes four theorems or principles, which have earlier been proven to be necessary conditions in order to avoid combinatorial effects (Mannaert et al., 2012 & 2016). These are '*separation of concerns*', '*data version transparency*', *action version transparency*, and '*separation of states*'. Applying these theorems leads to very fine-grained modular systems, which are very hard to create by human programming, as every violation of every principle leads to the introduction of a combinatorial effect. Therefore, five higher-level detailed design patterns (so-called elements) have been proposed to create NS software in practice (Mannaert et al., 2012, 2016), each aggregating and encapsulating a set of software constructs and providing the basic functionality of an information system. The five elements are '*data elements*', '*task elements*', '*flow elements*', '*connector elements*', and '*trigger elements*'. The elements have been proven to be free of combinatorial effects against a predefined set of anticipated changes (Mannaert et al., 2012 & 2016). Moreover, it has been argued that NS reasoning is applicable to modular architecture in general (De Bruyn, 2014; Mannaert et al., 2012 & 2016). More specifically, Van Nuffel (2011) illustrated that combinatorial effects could also be identified in business process models in the *'Business Process Modeling Notation'* (BPMN), and proposed guidelines on how they could be eliminated. Huysmans (2011) investigated the existence of combinatorial effects at the level of enterprise architectures, comprising both business- and IT-levels.

Design Structure Matrix

The Design Structure Matrix (DSM) has been proposed to map dependencies between elements in a system and address the complex issues related to '*Engineering'*, '*Product Design'*, '*Systems Science'*, '*Management Sciences'*, '*Organization Sciences'*, '*Healthcare Management'*, '*Public Policy'*, '*Natural Science'*, and '*Social Systems'* (Steward 1981; Eppinger, 1991; Baldwin & Clark, 2000; Eppinger & Browning 2012). In addition to being quite popular and being widely used in the field of Engineering and Product Design, recently the DSM has also been used in organization design. Out of ten chapters in their book, Eppinger and Browning (2012) devoted two chapters (4 & 5) on the application of DSM in organization architecture. Their effort suggests the significance of the concept of modularity at the organization level. Inter-modular and intra-modular dependencies can be identified by using the DSM while designing a product. An in-depth knowledge of the system and about the design parameters of inter-modular and intra-modular dependencies of different modules is the indispensable requirement in order to map a DSM. Once the process of designing systems architecture based on DSM is accomplished, each component (module) of the system becomes loosely-coupled, thereby facilitating outsourcing. For instance, Trapathy and Eppinger (2007, p. 21) state that *"The architecture-based DSM for PB's MEGA Midjet Series highlights how a product can be well partitioned by modules once the system architecture design has been completed. Such modular architecture can enable each module to be developed independently (out-shore/off-shore/in-house)"*. A set of design parameters of a product is plotted on a DSM (figure 2) reflecting that a particular parameter (i.e., 'Q1') is affected by other parameters of the design and also in turn, how the parameter 'Q1' is affecting other parameters of the design. In this way, a complete map of dependencies can be drawn in order to get the detailed structure of the product.

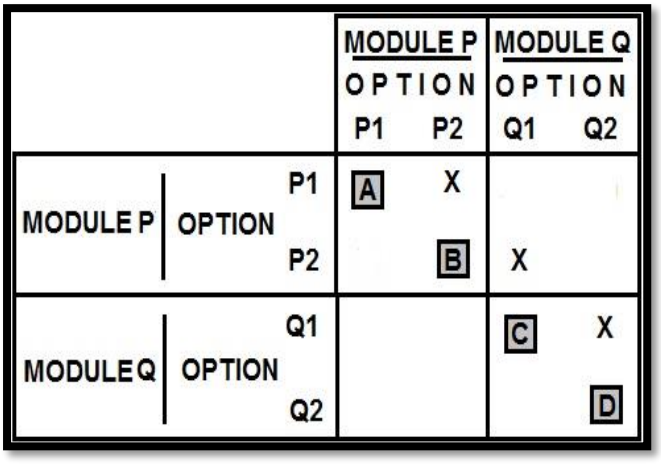

Figure 2. A simple DSM chart illustrated

How a simple DSM can be mapped, is illustrated in figure 2. In the DSM above (figure 2), 'X' is the dependency between two design options and the gray boxes (A, B, C, D) are the intersection points of identical design options. The first 'X' (design dependency) is at the intersection of column P2 and row P1. This 'X' explains that the design option of P2 influences the design option P1. In other words, the design decision for design option P1 is dependent on the design decision of design option P2. It means that there is a dependency between P1 and P2, but this dependency is not detrimental as it is occurring within a module (in this case module 'P'). This conforms to the statement made that the concept of modularity prescribes to have maximum interdependency within a module '*maximizing interdependence within them'* (Campagnolo & Camuffo, 2010).

Another example, the second 'X' (design dependency) is at the intersection of column Q1 and row P2. This 'X' explains that the design option of Q1 influences the design option P2. In other words, the design decision for design option P2 is dependent on the design decision of design option Q1. It means that there is a dependency between P2 and Q1, but this dependency is detrimental and considered as a violation of modular architecture as it occurs between the two modules (in this case module 'P' and module 'Q'). An indirect or chained dependency is also illustrated in figure 2. An indirect or chained dependency is a kind of dependency which does not influence directly but may have indirect influence. In the figure under reference, the third 'X' (design dependency) is at the intersection of column Q2 and row Q1. This 'X' explains that the design option of Q2 influences the design option Q1. In other words, the design decision for design option Q1 is dependent on the design decision of design option Q2. It means that there is a dependency between Q1 and Q2, but this dependency is not detrimental as it is occurring within a module (in this case module 'Q'). But the impact of this dependency may influence the design decision of design option P2 which is in another module (i.e. 'P') as P2 is influenced by the design decision of design option Q1 hence the design decision of Q2 may influence the design decision of P2 as well.

Baldwin and Clark's Modular Operators

Baldwin and Clark (2000) published a domain-independent theoretical framework on modularity in which they proposed six 'modular operators' to accommodate changes that can be applied in a modular system. These are: *splitting, substituting, augmenting, excluding, inverting, and porting*. Baldwin and Clark (2000) explain that operators are like actions that change the existing structure in order to improve a complex system. For instance, the inversion operator creates a new source of visible information (design rules) isolating the common features embedded in different modules; the porting operator allows making a module component compatible with other designs. Successful application of a modular operator is conditional on the assumption that no hidden modular dependencies are present in the system. These modular operators are further explained in the following sections.

***Splitting* a design (and its tasks) into modules**

The splitting operator is at the core of the modularization process because it permits to decompose a complex system to generate a set of loosely coupled modules. A fine-grained modular structure consists of many small modules instead of small sets of big modules. A fine-grained modular structure facilitates recombination of modules and offers greater flexibility.

- **Example**

An application module in software can be split into multiple smaller modules. There can be many ways of splitting an application module but one of those can be to split based on the package boundaries, for instance, a GUI module (graphical user interface), a data module, and a logic module. Usually, software's, to a certain extent, written following the principles of the modularity concept. The GUI module, data module, and logic module may probably be referred to as layers, each consisting out of a set of smaller modules.

***Substituting* one module design for another**

Substituting is also an important operator in order to design a modular system. *Substituting* explains a situation where a module can be replaced by another module which may have different characteristics.

- **Example**

The CRT monitor (Cathode ray tube) of a personal computer can be substituted by a flat panel monitor. Many components (modules) of a personal computer are substitutable as computers are designed and manufactured as a modular product.

***Augmenting/Adding* a new module to the system**

Adding a new module to the existing system will only succeed if the added new module does not hinder the good functioning of the existing system. A well-defined interface can easily facilitate adding modules without creating any coupling into the system.

- **Example**

Many old notebook computers were manufactured without having a camera or webcam. But augmenting a new module (webcam) to a notebook is very simple and this added new module will not create any coupling into the system. However, the operating system of the notebook should be compatible with the webcam (built-in driver) or if this is not the case a new driver has to be installed (augmenting at software level as well) which is compatible with the operating system. Personal computers are designed and manufactured as modular products; hence there should be no problem in augmenting one module.

***Excluding* a module from the system**

By applying the 'excluding' operator, the modular system can be minimized and if necessary, it can be incremented later on. In a well-designed modular structure, a module can be excluded without affecting the other modules.

- **Example**

For example, consider the internet as a modular system and the routers as modules. The internet is designed in such a way that if one router is excluded from the system, the system as a whole will continue to function. Thus, in such a situation exclusion of a module does not impact the system. The system as a whole remains operational as the packages will find another way to approach their destination.

***Inverting* to create new design rules**

In the context of modularity, inverting means that *"a system should not have many common elements in different modules"*. Inversion can be explained as *"collect common elements across several modules and organize them as a new level in the hierarchy"* (Baldwin & Clark, 2006). Modular operator '*Inverting*' explains that common modules in a system should be able to be consolidated in a single module which can move above in the hierarchy. In figure 3, modules 'A', 'B', and 'C' have a common element 'E'. The strategy is to identify common elements (E) from those modules (A, B, & C), and combine all those common elements together. Then create a new module (E) consisting of those common elements (E).

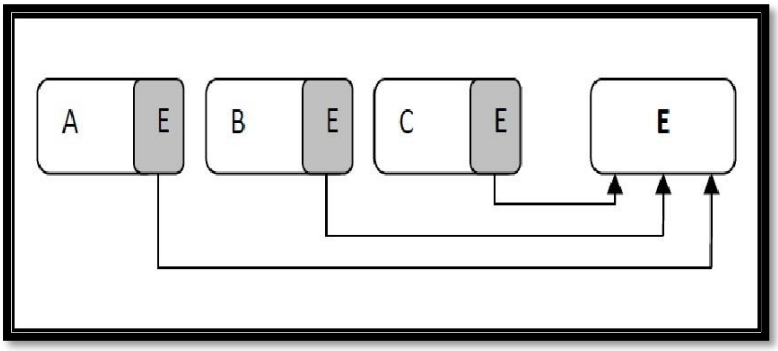

Figure 3. The Modular operator: Inversion

- **Example**

A printer driver is needed in many programs (i.e., word processing programs, drawing programs, database programs, etc.) to establish communication between a PC and a printer. The printer driver is not integrated individually in each of those programs; rather the printer driver is generalized and placed on a higher level of hierarchy in order to avoid duplication.

***Porting* a module to another system**

The modular operator porting facilitates moving a module from one system to another. The porting operator facilitates a module to become compatible with two separate systems.

- **Example**

Many old notebook computers are equipped with VGA (Video graphics array) output but HDMI (High definition multimedia interface) has become a *de facto* standard for almost all new flat screens. In order to connect a module (HDMI screen) to the system (notebook), there is a need to use an adapter (converter).

Findings

The findings of four analyzed cases answer the research questions. The findings illustrate several aspects of modularity that are relevant to IS/IT outsourcing projects (e.g., interface, modular architecture, cohesion, modular operator, coupling, dependencies, separation of concerns, design rules, standards, and encapsulation). It is noteworthy that, the relevance of the "modular architecture" aspect was found in three out of the four cases (table 2). Moreover, in these four cases, an indication of a positive correlation between the failure or difficulties in the projects and failure to meet the modularity requirements are also observed. In other words, while analyzing the cases applying the lens of modularity, violations in prescribed modularity requirements (as it is suggested by the modularity literature) were observed when a project was heading towards failure or going through difficulties. However, the authors by no means consider modular architecture a panacea and do not claim that the failure or difficulties are only due to these violations, but it is definitely a matter of interest to look at. Scholars and

practitioners alike will find good value in the insights uncovered and the lens of modularity might complement other tools/theories in order to avoid failures or difficulties in future IS/IT outsourcing projects. In the subsequent sections, three out of the four relevant cases are analyzed employing a systematic approach. The modular structure of the problem domain is made explicit first, and most importantly, the identification of modules is addressed. Second, the relevant modularity aspects are selected. Third, the resulting modularity requirements are listed, and fourth, the absence of modularity characteristics or requirements will be discussed in the context of violation or non-conformance of the modularity requirements.

Analysis: Introduction to cases

In the following, four cases are introduced briefly. It is noteworthy to mention that the third case proved to be irrelevant during the analysis of the modular architecture. Therefore, it has been excluded from this paper. The first case concerns the BSkyB vs. EDS outsourcing project. This case was of significant scale and complexity and ended in failure. The subsequent court case ended in 2010 and provided an exceptional amount of documentation. The re-analyses and results of this case study have been presented at the '*47th Hawaii International Conference on System Sciences'* (Huysmans et al., 2014) and a further elaborated analysis has been published in the '*International Journal of IT/Business Alignment and Governance'* (Huysmans et al., 2014).The second case concerns an IS/IT outsourcing project for a public university in a developing country in Asia. This IS/IT outsourcing project had a modest budget and scope, and was characterized by its simplicity and lack of complexity. Nevertheless, this project encountered significant issues and ended in failure. The findings of re-analysis have been published as a book chapter in the '*Encyclopedia of Information Science and Technology'* (Benazeer et al., 2018 & 2020). The third case pertains to one of the biggest service sector organizations in Belgium. Nevertheless, the outcomes of the empirical analysis based on the primary data failed to demonstrate the relevance of the 'modular architecture' aspect of the concept of modularity. Finally, the fourth case concerns a Belgian Financial Institution. In order to collect primary data, three investigators conducted multiple visits to the organization and multiple interview sessions with the CIO. This organization was involved in a multi-vendor IS/IT outsourcing project. The analysis of this case has been presented at the '*Enterprise and organizational Modeling and Simulation' (EOMAS 2017)* conference and later published in the *'Lecture Notes in Business Information processing: Enterprise and Organizational Modeling and Simulation'* (Benazeer, De Bruyn &Verelst, 2017). In the subsequent section, the analysis of the three cases is elaborated upon.

Analysis of Case 1 : The Role of Systems Integrator

This is a very complex case due to the size of the contract and also due to the inherent complexities of the project. The judgment runs over 2000 paragraphs, 468 pages, and took almost 18 months to deliver. While going through the qualitative data from different sources, some modular structures were identified in this case. So, the identified modular structures, specific modularity aspects, and specific requirements were outlined. When the actual situations were compared to the specific requirements, it has been found that in some circumstances, the actual situation does not correspond to the requirements; therefore, it can be considered as a violation or non-conformance of the requirement. For instance, the tasks of a systems integrator include staff selection, architecture design, scheduling, detailed design, and testing. In 2001, the overall state of the project was such that it became clear that:

> *"the CRM Project was in a poor state with no significant progress having been made in many areas"*
> (High Court of Justice, 2010, para: 1257).

Therefore, changes were made to the project approach. In March 2002, BSkyB took over the role of systems integrator from EDS (High Court of Justice, 2010, para: 27). In court proceedings (High Court of Justice, 2010, para: 470, 473, 477), it was stated that, as of a phone call on March 6, there was a consensus among the senior personnel of BSkyB and EDS that BSkyB would take over the role of systems integrator 'with immediate effect'. The exact terms of the take-over were determined in a '*Memorandum of Understanding*' on March 26.

i) Identifying the modular structure and requirements

The CRM project involved the integration of several legacy applications as well as some newly developed applications. Therefore, at the start of the project, BSkyB had not only selected EDS to develop the new CRM application, but also to perform the role of systems integrator. In this analysis, the role and responsibilities of a 'systems integrator' are defined as the system in scope and the modular structures at different levels conceived as modules. This configuration is referred to as '*modular structure A'.* The role of systems integrator, therefore, includes the responsibility to determine the modular architecture. The modular architecture comprises the definition of the modules which will make up the system, as well as the set of rules depicting the boundaries of design freedom for each module in order to let them properly work together as a whole. As all modules are required to adhere to the modular architecture, any modifications to that architecture may potentially have an impact on all modules, resulting in unpredictable change efforts. Therefore, the architecture should be designed at the beginning

of a modularization project, and should not be changed significantly afterward. For instance, Baldwin and Clark (2000, p. 127-128) argue that *"architectural changes do not arise naturally from incremental analysis [….]. The knowledge needed to make these judgments is gained from experience with prior designs, combined with a detailed understanding of the artifact's intended functions and of its physical logical structure"*. Therefore, the role and responsibilities of a 'systems integrator' in order to successfully manage a modular architecture require following steps. Firstly, to start with a clear definition of the modular architecture is indispensable. Secondly, changes in the modular architecture should not be made rashly or without thorough experience and expertise regarding the details of the design and prior history. In brief, the modular architecture should remain invariable and this requirement is referred to as '*modularity requirement A1'*.

ii) Assessing the modularity requirement

During the project, the role of systems integrator was changed, indicating that the initial modular architecture was perceived as inadequate. In the court proceedings, evidence can be discovered that demonstrate the impact of a systems integrator switch. For example, the selection of software packages to fulfill the functionality for a certain module as defined in the architecture, impacts other modules. One of the most important modules, the CRM module itself, was selected by the systems integrator: "*other integrators would have preferred the Siebel package instead of 'Chordiant'"* (High Court of Justice, 2010, para.1394). It has been acknowledged in the court proceedings that the change of systems integrator would not only have had an impact on the selection of packages, but would also have had a significantly impact on the integration between various modules (High Court of Justice, 2010, para: 1512).Although the CRM module itself was not changed when BSkyB assumed the role of systems integrator, other modules have been replaced. For instance, the 'Segue Silk' module was utilized for testing during phase 1 of the project; however, it was subsequently replaced by the 'Rock Solid' module following the assumption of BSkyB as a systems integrator. This change had a negative impact on the test automation, which delayed in implementation of the overall test strategy; and eventually the data migration to the new system (High Court of Justice, 2010, para: 1295). It is claimed that this is:

"…An example of the type of delay that a systems integrator who brings little experience of working with the products can encounter and which a systems integrator with good experience of the technologies can avoid" (High Court of Justice, 2010, para: 1299).

Moreover, BSkyB did not only change the pre-selected modules, but also attempted to change the application architecture and design:

"When the majority of coding had begun and was in test and it was too late to influence the design on which the coding was based" (High Court of Justice, 2010, para: 1741–1743).

BSkyB was aware of the large impact a change in systems integrator would have. This is illustrated by the consideration of appointing an alternative systems integrator, specifically one who is not affiliated with either EDS or BSkyB. However, it was concluded that such an appointment would:

"Push the delivery out too far" and that *"the cost would probably be significantly higher"*(High Court of Justice, 2010, para: 1415, 1423).

Nevertheless, the referenced examples demonstrate how, for long into the project, no stable modular architecture was achieved. From a modularity perspective, such project is difficult to end successfully. Indeed, it has been argued previously that (1) a good modular architecture should be decided on at the beginning of the project, (2) should preferably not be changed (optimized) in later stages of the project and (3) can certainly not be changed by another actor in an 'immediate 'way as it requires in-depth knowledge. As all these three points seemed to be lacking in the considerations of the project issues, hence the '*modularity requirement A1'* was not met. From this point, it could have been anticipated that a successful completion of the project was highly unlikely.

Analysis of Case 2 : Team Composition

In this case, the vendor organization is as referred to as '*Aries'* and the customer organization is referred to as '*Taurus*'. As required by the case organization, fictional names have been used in order to guarantee anonymity and confidentiality. '*Aries'* was a very competent and well-reputed provider as it was one of the leading independent companies working as a business unit of a large and reputed international company. '*Taurus'* was a big public-sector university in a developing country. The IS/IT project was to create a web-based portal for academic records management. The customer organization '*Taurus'* assigned a team of experts referred to as the 'focal team' who were responsible for communicating with vendor '*Aries'* and supervising each and every aspect of the project. Change is inevitable within organizations and accommodating change poses a challenge. It has been observed that

the composition of *'Taurus focal team'* changed three times over the course of the project (table 1). In this analysis, the non-technical root causes of failure are dealt with. The team's composition can be interpreted and explained in terms of modular structures (Huysmans et al., 2014). Furthermore, Terlouw (2011, p. viii) states, *"modules can comprise humans and/or software systems"* and in addition, Dietz (2006, p. 81) proposed a method to identify modular actor role structures and thereby asserts that *"an enterprise is constituted by the activities of actor roles, which are elementary chunks of authority and responsibility, fulfilled by subjects"*. As being said, the composition of the *'Taurus focal team'* changed and some members, including the team leader, were replaced by new members. To minimize the impact of changes within the *'Taurus focal team'*, management of '*Taurus'* appointed a software engineer to lead the *'Taurus focal team'* and expected him to coordinate with the '*Aries'*, up until the end of the project. Unfortunately, almost at the end of the project, during the testing phase, the team leader (a software engineer) also left for another job and the head of the department of computer science became the new team leader. Each change of *'Taurus focal team'* composition caused some kind of hindrance to the IS/IT project.

N°	TEAM COMPOSITION - TAURUS FOCAL TEAM
1	The 'Taurus Focal Team' comprises senior faculty members from different departments. The team leader was one of the heads of departments (1st team leader).
2	Due to sme routine and policy decisions, some team members were replaced by new team members.
3	A software engineer was appointed as a new team leader (2nd team leader).
4	During the testing phase of IS/IT project, the 2nd team leader departed for another job. Subsequently, he was substituted by the 'Head of the Computer Science Department' (3rd team leader).

Table 1. Different compositions of the Taurus focal team.

i) Identifying the modular structure and requirements

The modularity literature suggests that within a system modular architecture can be studied at different levels. The identity of any unit as a modular system is not fixed. This identity is determined by the level of analysis chosen within a system (Schilling, 2000; Simon, 1962). If an industry is considered a system, then a unit of that system, for instance, an organization, can be considered a module of that system. By further narrowing the scope and considering an organization as a system, various departments (e.g., production, marketing, human resource, etc.) within that system can be regarded as modules. Employing this analogy, the organization '*Taurus'* as a system can be studied at three distinct fine-grained levels. The organization '*Taurus'* is considered as the first level of a module, which can be further subdivided into two (hierarchically nested) smaller levels (i.e. sub-modules and sub-submodules). A first sub-modular level refers to the *'Taurus focal team'*, and this configuration is referred to as *modular structure B1*. The second sub-submodular level refers to *'an individual team member'*, for instance, a member of the '*Taurus focal team*'. This configuration is referred to as *modular structure B2*. In this section, the analysis focused on two levels of modular hierarchy. Firstly, the analysis B1 starts with sub-modular level (*Taurus focal team*) and the focus of the analysis was on the *'substitution'* aspect, a modular operator (Baldwin & Clark, 2000). Secondly, the analysis B2 goes down to the next hierarchical level, i.e., sub-submodular level (an individual member of *'Taurus' focal team*) and the focus of the analysis was on the *'modular architecture'*, an aspect of the concept of modularity.

Baldwin and Clark (2000, p. 262) suggest that *"The substitution operator allows a designer (or user) to swap one module of the system for a better version of the same module"*. In a well-designed modular IS/IT system, applying the modular operator '*substitution*' should not impact the existing structure negatively. Modularity literature suggests that *"substituting an older version of the module with the newer version should ameliorate the overall performance of the system"* (Huysmans et. al., 2014, p. 4418). Terlouw (2011, p. viii) asserts that *"the modular operators are the actions that may change existing structures in a well-defined way in order to enhance the efficiency of the system"*. The modular operator '*substitution*' can be applied successfully and relatively easily if all module versions adhere to the same interface and no undocumented inter-modular dependencies are present. If the interface is changed, and/or the dependencies of the modules are not made explicit, the application of the modular operator *'substitution'* is not without risk. One such risk is that applying the *'substitution'* operator disrupts the working of the system and may trigger ripple effects. So, according to the concept of modularity in a

well-designed modular architecture, the modular operator *'substitution'* can be applied relatively easily in order to ameliorate the overall performance of the system and this condition is considered as *modularity requirement B1.1*.

Moreover, when considering a team (*Taurus focal team*) as a modular structure, subsequently the boundaries of the authority of each team member can be considered as an interface. Consequently, these boundaries should be made explicit and should not change when the module is substituted. Restricting or freezing the boundaries of each module will facilitate to identify any hidden dependencies between the modules and also safeguard the integrity of the modular architecture. As it has been mentioned earlier, a modular architecture can be described as *"a predefined set of prescriptive rules which all modules of the system need to adhere to"* (Huysmans et al., 2014, p. 4418). This definition clearly indicates that once the modular architecture is designed, the modules (*members of the 'Taurus focal team'*) are obliged to adhere to the predefined rules. The modules should remain within their defined boundaries and should not overstep it. So, the boundaries of the modules (*members of the 'Taurus focal team'*) should be defined clearly and the integrity of the modular architecture should remain unchanged and this condition is considered as *modularity requirement B2.2*.

ii) Assessing the modularity requirement

In this case, it has been observed that the composition of the sub-module *'Taurus focal team'* has changed three times (table 1). Using modularity terminology, this means that the sub-module has been substituted three times. Initially, the leader of the focal team only acted as a facilitator in order to communicate effectively and efficiently with the vendor '*Aries'*. Therefore, the team leader of *'Taurus focal team'* has no authority to make any changes to the project conditions which were already agreed upon by all the stakeholders. In reality, this was obviously not the case. Whenever another version of the *'Taurus focal team'* was put in place, already established agreements between both parties tended to change (i.e., requirements specification was developed and agreed upon by the customer and the vendor). Substituting sub-module *'Taurus focal team'* with newer versions was negatively affecting the efficiency of the project which can be observed by the following excerpts:

"Changes at the organizational level [….] led to some new requirements emerging from nowhere and caused frequent changes in the old requirements [….]"(Nauman, Aziz, & Ishaq, 2009, p. 270).

The above excerpt illustrates that applying the modular operator *'substitution'* resulted in significant problems, therefore, in this situation and in the context of this case, it can be concluded that *modularity requirement B1.1* was not met.

Moreover, it has been observed that frequent changes in requirements are always enacted by the incoming new members of the *'Taurus focal team'*. The scope and objectives of the project were already agreed upon by the customer organization '*Taurus*' and the vendor organization '*Aries'* at the beginning of the project. But the incoming new members probably ignored the predefined scope and objectives of the project and by doing this, the sub sub-modules (*members of the focal team*) did not remain within their defined (allocated) boundary. This led to the loss of integrity of the modular architecture and it can be observed by the following excerpt:

"The new members of the focal team were not clear about the scope and objectives of the project [….]. Due to this kind of divisive environment, a huge time was lost in the advancement of the project" (Nauman, Aziz, & Ishaq, 2009, p. 270).

According to the concept of modularity, sub sub-modules (*members of the 'Taurus focal team'*) were supposed to be free of any hidden dependencies by restricting or freezing the boundaries of each sub sub-module in order to safeguard the integrity of the modular architecture but unfortunately this did not happen therefore, it can be concluded that *modularity requirement B2.2* was not met.

Analysis of Case 3 : Irrelevant to the 'Modular Architecture'

The third case pertains to one of the biggest service sector organizations in Belgium. However, the findings of the analysis fail to demonstrate the relevance of the 'modular architecture' aspect of the concept of modularity.

Analysis of Case 4 : Fine-Grained Modular Architecture

The fourth and final case of this study concerns a Belgian financial institution (AB bank) that employs a multi-vendor outsourcing approach. As required by the case organization, fictional names have been used in order to guarantee anonymity and confidentiality. The main activity of the in-house IT team of the financial institution was therefore concerned with the integration of all outsourced activities as well as its general management (package selection, vendor negotiations, etc.). The outsourcing contracts (SLA's) need to be managed by good arrangements stipulating the roles and responsibilities of each of the involved actors as these deals are often highly complex and of crucial importance for both parties. First, it is clear that such SLA is crucial from a legal point of view. Numerous IS/IT outsourcing projects fail and may result in non-satisfactory relationships between the vendor and the customer which may sometimes even end up in a legal dispute (e.g., case 1). In such cases, obviously, the SLA

serves as the starting point to analyze who has (not) fulfilled his or her responsibilities. However, the architecture of an SLA (interface) holds immense significance. Based on literature, it can be argued that it is imperative to be precise regarding the set of individuals, information, and rules governing the flow of information between the parties. For instance, interfaces in services can include people, information, and rules governing the flow of information (Voss & Hsuan, 2009). At the industry level, these interfaces often consist of regulatory frameworks, rules, standards, and technical specifications that allow different players to connect (Jacobides, Knudsen, & Augier, 2006). Modularity is inherently a recursive concept that can be applied at different levels. This analysis revealed two major levels at which modularity could be clearly applied to the case at hand (i.e., inter-organizational level and intra-organizational level). Due to space limitations, this analysis will focus only on the inter-organizational level.

i) Identifying the modular structure and requirements

In this analysis, the SLA itself conceived as the system and the different clauses, rules or paragraphs as the modules within that system. Indeed, one could consider documents and legal documents in particular, as modular systems (Blair, O'Connor, & Kirchhoefer, 2011; Smith, 2006). The paragraphs, items or clauses within such documents are clearly identifiable units which are often reused within different contracts and are aggregated into one specific contract acting as the specific SLA for a specific outsourcing agreement. This configuration is referred to as *modular structure C.* However, given a long-term duration of many outsourcing contracts, it is conceivable that mutually agreed upon changes in the SLA do need to happen as often changing circumstances emerge (e.g., additional or improved services from the vendor to the customer). As this kind of change of interface happens purposefully in order to accommodate the changing circumstances, this should not be a problematic issue. However, in order to enable the agility and flexibility of the SLA contract, it is required that the SLA itself consists of smaller parts or modules (*fine-grained modular architecture*). Each of these individual contract modules can then be updated or removed from the contract, or additional ones can be added. So, there is a need to draft a fine-grained SLA contract which can be adopted at the level of individual clauses and this statement is referred to as *modularity requirement C1.*

ii) Assessing the modularity requirements

Discussions arose with the vendors on who should undertake which tasks and it was highlighted that *'AB Bank'* was unable to insist on getting the expected services performed properly due to their limited leverage over the respective vendors. When asked about the possibility to adapt the SLAs with its vendors during the execution of an agreement, the informant indicated that such things simply did not happen at *'AB Bank'*. The informant also added that this usually assumed to be covered by mutual trust:

> "Let's say tomorrow if I approach one of my suppliers and request him for a change. If they come back and inform me that they do not want to change it, there are no obligations to change things on my request. No, we do not have any such guarantee. [….] You may take it for granted that they will help you in making changes you need".

The IS/IT outsourcing contracts at *'AB Bank'* are often negotiated and drafted as a whole, and there is no established procedure for accommodating changes. Contracts are typically signed for a fixed duration, for instance, five years, and therefore should be renegotiated within the borders of this cycle. The only type of change which was explicitly taken into account was the (premature) termination of the contract initiated by the client (i.e., *'AB Bank'*). This was mainly considered from the financial and legal perspective, such as stipulations regarding the maximum amounts of costs for *'AB Bank'* in order to quit the deal and enable to switch to another vendor, as in the following the informant stated:

> "In that contract, we had a very good clause detailing the processes when we want to exit. How do we have to start it, what are the processes to follow, what are the steps that we need to take? [….] It was stated in the clause that the exit will not cost more than the amount we paid for the setup".

From these findings, it is evident that the SLA was considered to be one monolithic block that was to be dealt with or re-negotiated, contrary to the ideally prescribed modular structures aimed at change and flexibility during the stipulated duration of the contract. Therefore, it can be concluded that *modularity requirement C1* was not met.

CASE	MODULAR STRUCTURE	MODULARITY ASPECT	MODULARITY REQUIREMENT	CONFORMANCE
1	SYSTEMS INTEGRATION *Modular structures at different levels are conceived as modules*	Modular architecture	Modular architecture should remain invariable	Not met
2	THE CUSTOMER ORGANIZATION *The organization 'Taurus' is conceived as a module and the 'Taurus focal team' is conceived as sub-module. In analogy, every member of the 'Taurus focal team' is conceived as (sub) sub-module*	Modular operator 'substitution' --- Modular architecture	The substitution operator can be applied successfully and relatively easily if all inter-modular dependencies are explicit. The boundaries of the modules should be defined clearly, and the integrity of the modular architecture should remain unchanged	Not met
3	Failed to demonstrate the relevance of the 'modular architecture' aspect of the concept of modularity	Modular architecture	Failed to demonstrate the relevance of the 'modular architecture' aspect of the concept of modularity	
4	SERVICE LEVEL AGREEMENT (SLA) *The paragraphs, items or clauses within an SLA are conceived as modules*	Modular architecture	Fine-grained	Not met

Table 2. The summary of the findings

The above table (2) summarizes the findings from the three cases analyzed above. The first column 'modular structure' refers to the focus of the modular structure considered during analysis. The use of capital letters denotes the 'system', while the use of italic letters denotes the modules of this system. The second column 'modularity aspects', indicates which aspect of modularity was the most suitable for this analysis In this case, 'modular architecture' aspect was examined. The third column describes the requirements for this particular aspect as they are described in the modularity literature. The last column illustrates the outcomes, indicating whether the actual situation was conformed to the modularity requirements or not. The subsequent section conducts a comprehensive analysis of four cases, employing the cross-case analysis method, and examines the relevance of modularity aspects as depicted in table 3.

Cross Case Analysis

In addition to a thematic analysis, a cross-case analysis is also conducted in order to enhance the external validity. In the following table (3), a cross-analysis using replication logic approach of four cases has been done. In this analysis it has been observed that some modularity aspects have emerged multiple times illustrating the level of relevance in analyzed four cases. For instance, the 'modular architecture' as the most relevant modularity aspect has emerged in three out of the four cases, following the 'interface' aspect.

Modularity aspects	Case 1	Case 2	Case 3	Case 4	Result
Interface / SLA	X	X	X	X	4
Modular architecture	X	X		X	3
Cohesion	X			X	2
Modular operator		X	X		2
Coupling			X		1
Dependencies		X			1
Separation of concerns	X				1
Design rules		X			1
Standards					0
Encapsulaion					0

Table 3. Cross case analysis

Some other relevant modularity aspects, for instance, 'cohesion' and 'modular operator' both have emerged in two cases and the least relevant modularity aspects 'design rule', 'dependencies', 'coupling', and 'separation of concerns' each have emerged in one case. The authors further like to clarify that the level of relevancy is a context specific factor. The above illustrated results (table 3) are relevant in the context of these four analyzed cases. However, the order of relevance may differ in different context. The contribution of this cross-case analysis is at least it facilitates a deeper insight and understanding about the weight attributed to each aspect.

Conclusion

IS/IT project outsourcing has become one of the most important developments in the IT sector over the past decades. Notwithstanding the impressive results that have been achieved, research indicates that many IS/IT outsourcing projects do not deliver the expected results. In order to gain insight into this, this research investigates whether and how the concept of modularity can be applied to IS/IT project outsourcing. The necessity for applying the concept of modularity was prompted by the mixed results of IS/IT outsourcing projects in practice, as well as the research on Normalized Systems, which has demonstrated predominantly at the software level, that the evolvability of modular structures is limited by a form of coupling called combinatorial effects, which affects not only evolution but many aspects of development of modular structures as well. This study is a first attempt to gain insight into whether these modularity aspects studied in Normalized Systems affect IS/IT outsourcing as well. The instances or examples of modularity observed in four analyzed cases are listed in table 3. It is not surprising to find that in three out of the four cases, the 'modular architecture' related issues were observed. The results obtained from the aforementioned analyzed cases provide instances and examples of modularity, enabling the identification of pertinent modularity aspects in the context of IS/IT project outsourcing. Furthermore, the findings have become more robust due to the identification of the relevance of modularity observed at two levels of IS/IT projects; technical and organizational. For instance, the relevance of modularity at the technical levels is identified in case 1, whereas cases 2 and 4 elucidate the relevance of modularity at the organizational levels. These identifications demonstrate the relevance of modularity in the context of IS/IT project outsourcing, thereby answering the first research question: "*Which instances (examples, counter-examples) of the use of modularity in the context of IS/IT outsourcing can be identified"?* These identified instances or examples may be associated with the theoretical frameworks or design principles that may influence the success or failure of an IS/IT outsourcing project. Furthermore, the identified instances or examples in the analyzed cases may be construed as contraventions of, or at the very least inadequate consideration of, the design principles prescribed by the concept of modularity. Thereby this provides evidence that these failures are likely to have a negative impact on the complexity of the project and ultimately influence the success or failure of the project. These findings answer the second research question: *"How can the relevance and/or importance of these instances (examples, counter examples) for IS/IT outsourcing project be assessed"?* Finally, the authors would like to emphasize that the present study does not imply that modularity is the sole or predominant factor determining the success of IS/IT outsourcing projects. Instead, the objective of this study was to examine the relevance of a factor that, in the author's perspective, is often underexposed in the context of IS/IT project outsourcing, without diminishing the significance of other factors. This study has provided indications of the role that modularity plays in IS/ITproject outsourcing, in a wide range of domains, from organizational to technical, with implications on areas such as knowledge management and communications. Even though this study was not specifically aimed at practitioners, however, it does offer several guidelines for practitioners in IS/IT project outsourcing. This study highlights the complexity involved in IS/IT outsourcing projects, based on the concept of modularity. The present study further illustrate that, even in IS/IT outsourcing projects of moderate size, not only complex modular structures can be present at the technical level (i.e., software), but also at the organizational level (i.e., teams, departments, SLA's, business processes, documents, etc.). To summarize, the findings of the analyses indicate that the concept of modularity seems to be certainly relevant in the context of IS/IT project outsourcing.

References

Baldwin, C. Y., & Clark, K. B. (2000). *Design rules: The power of modularity*. The MIT Press.

Baldwin, C. Y., & Clark, K. B. (2006). Modularity in the design of complex engineering systems. In D. Braha, A. A. Minai, & Y. Bar-Yam (Eds.), *Complex engineered systems: Science meets technology*. Springer.

Bapna, R., Gupta, A., Ray, G., & Singh, S. (2013). *Specialization, integration, and multi-sourcing: A study of large IT outsourcing projects* [Paper presentation]. International Conference on Information Systems. Milan, Italy.

Benazeer, S., De Bruyn, P., & Verelst, J. (2017). Applying the concept of modularity to IT outsourcing: A financial services case. In R. Pergl, R. Lock, E. Babkin, & M. Molhanec (Eds.), *Enterprise and organizational modeling and simulation* (pp. 68-82). Springer.

Benazeer, S., Huysmans, P., De Bruyn, P., & Verelst, J. (2018). The concept of modularity and normalized systems theory in the context of IS outsourcing. In M. Khosrow-Pour (Ed.), *Encyclopedia of information science and technology* (pp. 5317-5326). IGI Publications.

Benazeer, S., De Bruyn, P., & Verelst, J. (2020). The concept of modularity in the context of IS/IT project outsourcing: An empirical case study of a Belgian technology services company. *International Journal of Information System Modeling and Design, 11*(4), 1-17.

Benbasat, I., Goldstein, D. K., & Mead, M. (1987). The case research strategy in studies of Information Systems. *MIS Quarterly, 11*(3), 369-386.

Blair, M. M., O'Connor, E. O., & Kirchhoefer, G. (2011). Outsourcing, modularity, and the theory of the firm. *BYU Law Review, 2011*(2), 262-314.

Campagnolo, D., & Camuffo, A. (2010). The concept of modularity in management studies: A literature review. *International Journal of Management Reviews, 12*(3), 259-283.

Colfer, L., & Baldwin, C. Y. (2016). *The mirroring hypothesis: Theory, evidence and exceptions*, (Finance Working Paper No. 16-124). Harvard Business School. Accessed on 23/01/2024, retrieved from: SSRN: https://ssrn.com/abstract=2770675.

De Bruyn, P. (2014). *Generalizing normalized systems theory: towards a foundational theory for enterprise engineering*. [Doctoral dissertation, University of Antwerp, Belgium].

Delens, G. P. A. J., Peters, R. J., Verhoef, C., van Vlijmen, S. F. M. (2016). Lessons from Dutch IT-outsourcing success and failure. *Science of Computer Programming, 130*(32), 37-68.

Dietz, J. L. G. (2006). *Enterprise ontology: Theory and methodology*. Springer.

Eppinger, S. D. (1991). Model-based approaches to managing concurrent engineering. *Journal of Engineering Design, 2*(4), 283-290.

Eppinger, S. D., & Chitkara, A. R. (2009). The practice of global product development. MIT *Sloan Management Review*, July 1.

Eppinger, S. D., & Browning, T. R. (2012). *Design structure matrix methods and applications*. The MIT Press.

Greaver II, M. F. (1999). *Strategic outsourcing: A structured approach to outsourcing decisions and initiatives*. Amacom.

Harris, M. D. S, Herron, D., & Iwanicki, S. (2008). *The business value of IT: Managing risks, optimizing performance and measuring results*. CRC Press.

High Court of Justice. (2010). BSkyB limited, sky subscriber services limited, HP enterprise services UK limited. London, UK: Electronic data systems LLC.

Huysmans, P. (2011). *On the feasibility of normalized enterprises: Applying normalized systems theory to the high-level design of enterprises*. [Doctoral dissertation, University of Antwerp, Belgium].

Huysmans, P., De Bruyn, P., Benazeer, S., De Beuckelaer, A., De Haes, S., & Verelst, J. (2014). *On the relevance of the modularity concept for understanding outsourcing risk factors* [Paper presentation]. Proceedings of the 47th Hawaii International Conference on System Sciences. Hawaii, USA.

Huysmans, P., De Bruyn, P., Benazeer, S., De Beuckelaer, A., De Haes, S., & Verelst, J. (2014). Understanding outsourcing risk factors based on modularity: The BSkyB case. *International Journal of IT/Business Alignment and Governance, 5*(1), 50-66.

Ishizaka, A. & Blakiston, R. (2012). The 18C's model for a successful long-term outsourcing arrangement. *Industrial Marketing Management, 41*(7), 1071-1080.

Jabangwe, R., Smite, D., & Hesbo, E. (2016). Distributed software development in an offshore outsourcing project: A case study of source code evolution and quality. *Information and Software Technology, 72*, 125-136.

Jacobides, M. G., Knudsen, T. R., & Augier, M. (2006). Benefiting from innovation: Value creation, value appropriation, and the role of industry architectures, *Research Policy, 35*(8), 1200–1221.

Karimi-Alaghehband, F., & Rivard, S. (2012, December). *Information technology outsourcing success: A model of dynamic, operational, and learning capabilities* [Paper presentation]. 33rd International Conference on Information Systems (ICIS). Florida, USA

Kruchten, P., Obbink, H., & Stafford, J. (2006). The past, present, and future for software architecture. *IEEE Software 23*(2), 22-30.

Lambert, D. M., Emmelhainz, M. A., & Gardner, J. T. (1999). Building successful logistics partnerships. *Journal of Business Logistics, 20*(1), 165-181.

Lee, J-N. (2001). The impact of knowledge sharing, organizational capability and partnership quality on IS outsourcing success. *Information & Management, 38*(5), 323-335.

Lehman, M. M. (1980). Programs, life cycles, and laws of software evolution. *Proceedings of IEEE, 68*(9), 1060-1076.

Logan, M. S. (2000). Using agency theory to design successful outsourcing relationships. *The International Journal of Logistics Management, 11*(2), 21-32.

MacDuffie, J. P. (2013). Modularity-as-property, modularization-as-process, and 'modularity' as frame: Lessons from product architecture initiatives in the global automotive industry. *Global Strategy Journal, 3*(1), 8-40.

Mannaert, H., Verelst, J., & Ven, K. (2012). Towards evolvable software architectures based on systems theoretic stability. *Software practice and experience, 42*(1), 89-116.

Mannaert, H., Verelst, J., & De Bruyn, P. (2016). *Normalized systems theory: From foundations for evolvable software toward a general theory for evolvable design*. Koppa.

Modarress, B., Ansari, A., & Thies, E. (2014). The impact of technology transfer through foreign direct investment in developing nations: A case study in the United Arab Emirates. *International Journal of Economics & Finance, 6*(7), 108-126.

Nagpal, P., & Lyytinen, K. (2010, December). *Modularity, information technology outsourcing success, and business performance* [Paper presentation]. International Conference on Information Systems. St. Louis, USA.

Nauman, A. B., Aziz, R., & Ishaq, A. F. M. (2009). Information system development failure and complexity: A case study. In M. G. Hinter (Ed.), *Selected reading on strategic information systems* (pp. 251-275). IGI Publications.

Peterson, B. L., & Carco, D. M. (1998). *The smart way to buy information technology: How to maximize value and avoid costly pitfalls*. Amacom.

Rottman, J. W. (2008). Successful knowledge transfer within offshore supplier networks: A case study exploring social capital in strategic alliances. *Journal of Information Technology, 23*(1), 31-43.

Sako, M. (2005). Modularity and outsourcing. In A. Principe, A. Davies & M. Hobday (Eds.), *The business of system integration* (pp. 229-253). Oxford University Press.

Sako, M. (2014). Outsourcing and offshoring of professional services. In L. Empson, D. Muzio, J. Broschak, & B. Hinings (Eds.), *The oxford handbook of professional service firms* (pp. 327-347). Oxford University Press.

Sanchez, R., & Shibata, T. (2021). Modularity design rules for architecture development: Theory, implementation, and evidence from the development of the Renault–Nissan alliance "common module family" architecture. *Journal of Open Innovation: Technology, Market, and Complexity, 7*(4), 1-22.

Schilling, M. A. (2000). Towards a general modular systems theory and its application to inter-firm product modularity. *Academy of Management Review, 25*(2), 312-334.

Schmidt, N., Zoller, B., & Rosenkranz, C. (2016). The clash of cultures in information technology outsourcing relationships: An institutional logics perspective. In J. Kotlarsky, I. Oshri, & L. P. Willcocks (Eds.), *Shared services and outsourcing: A contemporary outlook* (pp. 97-117). Springer.

Simon, H. A. (1962). The architecture of complexity. *Proceedings of the American Philosophical Society, 106*(6), 467-482.

Smith, H. E. (2006). Modularity in contracts: Boilerplate and information flow. *University of Michigan Law Review, 104*, 1175-1222.

Steward, D. V. (1981). The design structure system: A method for managing the design of complex systems. *IEEE Transactions on Engineering Management, 28*(3), p. 71-74.

Terlouw, L. I. (2011). *Modularization and specification of service-oriented systems* [Doctoral dissertation, Delft University of Technology, The Netherlands]. Gildeprint.

Trapathy, A., & Eppinger, S. D. (2007). *A system architecture approach to global product development*, (Working Paper Number 4645-07). MIT Sloan School of Management.

Van Nuffel, D. (2011). *Towards designing modular and evolvable business processes*. [Doctoral dissertation, University of Antwerp, Belgium].

Voss, C. A., & Hsuan, J. (2009). Service architecture and modularity. *Decision Sciences, 40*(3), 541-569.

Wojewoda, S., & Hastie, S. (2015). *Standish group 2015 chaos report - Q&A with Jennifer Lynch*. Infoq. Retrieved May 2, 2024, from https://www.infoq.com/articles/standish-chaos-2015.

Yin, R. K. (2014). *Case study research: Design and methods*. Sage Publications.

Zheng, Y., & Abbott, P. (2013, June). *Moving up the value chain or reconfiguring the value network? An organizational learning perspective on born global outsourcing vendors* [Paper presentation]. Proceedings of the 21st European Conference on Information Systems. Utrecht, The Netherlands.

Zirpoli, F., & Becker, M. C. (2011). The limits of design and engineering outsourcing: Performance integration and the unfulfilled promises of modularity. *R&D Management Review, 41*(1), 21-43.