Lebanese American University Repository (LAUR)

Post-print version/Author Accepted Manuscript

Publication metadata

Title: Innovative sustainable methodology for managing in-house software development in SMEs

Author(s): Abbas Tarhini, Manal Yunis, Abdul-Nasser El-Kassar

Journal: Benchmarking: An International Journal

DOI/Link: https://doi.org/10.1108/BIJ-05-2017-0103

How to cite this post-print from LAUR:

Tarhini, A., Yunis, M., & El-Kassar, A. N. (2018). Innovative sustainable methodology for managing in-house software development in SMEs. *Benchmarking: An International Journal*., DOI, 10.1108/BIJ-05-2017-0103, http://hdl.handle.net/10725/11188

© Year 2018

© 2018 Emerald Publishing Ltd. This AAM is provided for your own personal use only. It may not be used for resale, reprinting, systematic distribution, emailing, or for any other commercial purpose without the permission of the publisher'.

This Open Access post-print is licensed under a Creative Commons Attribution-Non Commercial-No Derivatives (CC-BY-NC-ND 4.0)

This paper is posted at LAU Repository

For more information, please contact: archives@lau.edu.lb

Paper Title: Innovative Sustainable Methodology for managing in-house Software Development in SMEs

Journal: Benchmarking: An International Journal. H Index 54, Rank: Q2

Citation: Tarhini, A., Yunis, M., El Kassar, A. (2018). Innovative Sustainable Methodology for managing in-house Software Development in SMEs. Benchmarking, 25(3), pp. 1085-1103.

Link: https://www.emerald.com/insight/content/doi/10.1108/BIJ-05-2017-0103/full/html

Innovative Sustainable Methodology for managing in-house Software Development in SMEs

Abbas Tarhini
abbas.tarhini@lau.edu.lb

Manal M. Yunis
myunis@lau.edu.lb

Abdul-Nasser Kassar
abdulnasser.kassar@lau.edu.lb

Department of Information Technology and Operations Management,
Lebanese American University, Beirut, Lebanon

Abstract

Purpose – This paper presents an innovative agile methodology that proposes fundamental changes in managing the development of in-house information systems in SMEs and benchmarks it with one of two database technologies enabling these systems to be both efficient and competitive

Design/methodology/approach – The objectives are achieved by presenting an elaborated design of the agile methodology that manages the system development process by addressing three basic components: Roles played by system players, Process needed to fulfil the system development, and Artifacts to document the project. A case study is conducted as a proof of the effectiveness of the proposed methodology and measures whether the selection of the database technology affects the effectiveness of the system development process.

Findings – Results show that, compared with traditional methodologies, the proposed methodology reduced the cost of system development and testing by 30% and enhanced the IT – business alliance. Further, this work found that the selection of a suitable database technology is strongly related to the complexity and interrelationships between the data used.

Originality/value – Such research did not receive the needed attention (Hunter, 2004) even in the past decade. Successful adoption of IT by companies could be in the form of customized IS which could be expensive for SMEs to adopt due to a lack in technical expertise and financial resources. The proposed methodology has the potential to promote sustainable development through helping SMEs in reducing the time and cost of IT project development.

Keywords- SMEs; In-house Application Development; Agile Methodology; Relational Database; RDB; ORDB

Paper type- Research paper/Case study

1. Introduction

Years ago business organizations became aware of the need for IT in implementing their business strategies and achieving their competitive advantage; yet IT usage posed a paradox in results. On one hand, according to a report by the *Economist*, the spread of computers will increase these associated emissions by about 6% a year until 2020, when the adoptions of personal computers, mobile phones, and broadband internet connection get remarkably higher (Economist, 2009). On the other hand, small and medium-sized businesses could reduce electrical power costs by shifting to distributed computing, rather than adopting and operating their own servers. The delivery of computer services via clouds, instead of massive sets of shared machines, enables firms to outsource the running of their applications, accounting systems, and customer databases to someone else. Another computing model is virtualization—the creation of "virtual" machines so that multiple operating systems and applications can run on the same physical computer. Sun Microsystems reports that 70% of the servers in most organizations have only one application running on them. Consolidating these applications onto fewer machines would thus be greener and more efficient. Companies adopting such computing models achieve higher efficiency levels in IT usage, and thus reduce not only their computing costs but also their carbon emissions (O'Neill, 2010; Raisinghani & Idemudia, 2015).

Still another dilemma is that of insourcing or outsourcing. Several organizations find that building their own applications allow them to customize the functionalities of each process according to their unique requirements (Haider, Samdani, Ali, & Kamran, 2016), and to control the quality of their application (Eskelin, 2001). On the other hand, other studies highlight on the advantages of outsourcing over in-house development in terms of flexibility, cost and time reduction (Meyers & Oberndorf, 2001; Dogerlioglu, 2012). Still other researchers argues that several firms, especially small and medium-sized enterprises (SMEs), choose in-house development since it allows them to control the cost of development and to deal with changes in the company and environment (Moody & Walsh, 1999; Olsen, 2011). However, this cost control is only possible if the project went as smooth as it is planned with no scope and budget creep; such scope and budget creep cannot be avoided if the project development methodology and its physical implementation are not well selected (Babu, 2005; Newton, 2015). Unfortunately, small and medium sized organizations lack the adoption of well-established methodologies of project management or experienced developers who can apply methodologies adopted in large enterprises (Ilincuta & Jergeas, 2003). In fact, today small and medium enterprises need customized information systems that address their business processes; however these SMEs lack technical expertise, and have a limited financial abilities; and thus they cannot adopt well-established methodologies as large enterprises, so they decide to develop in-house projects using existing limited resources ignoring current project management standards (Rowe, 2007) which is unwelcomed by SMEs as they perceive it bureaucratic and complicated in terms of time and budget constraints (PMI, 2004). The success of such projects requires the correct selection of the appropriate technologies related to databases and the programming environment (Thomas & Begg, 2001), and also is related to the system development methodologies that help to accomplish project

objectives with the least cost (Chin, 2003). As for database technologies, in addition to Relational Databases (RDB), the Object Relational Database (ORDB) technology is becoming very popular nowadays and is seen as the next generation of the database systems (Thomas & Begg, 2001; Kim & Lochovsky, 1989; Silberschatz, Korth, & Sudarshan, 2009); still ORDB is criticized because of its abstract representation of tables which is viewed as a complexity over the simple representations in the relational database model (Wang, 2010). As for the software development methodologies, the agile methods are perceived as promising methodologies because they target the current challenges and problems of SMEs (Jin-Hai, Anderson, & Harrison, 2003; Alshayeb & Li, 2005; Vazquez-Bustelo & Avella, 2006). Further, these methods are not process-centered; however, they manage the unpredictable people actions which are shaped by their culture and behavior (Cockburn & Highsmith, 2001). Agile methodology is an alternative to complex methods, and has gained significant acceptance within small development teams (Schwaber, 2004).

Today, enriching the research related to IT implementations in SMEs is essential as such research did not receive the needed attention (Hunter, 2004) even in the past decade; this is because of research problems related to project sizes, and lack of organization resources to support such research (Hunter, 2004). Therefore, such kind of research should be supported, as SMEs has a significant effect on economies, especially in developing countries. According to the World Bank report in 2015 (Tima, 2015), "formal SMEs contribute up to 45 percent of total employment and up to 33 percent of national income (GDP) in emerging economies. These numbers are significantly higher when informal SMEs are included. According to estimates, 600 million jobs will be needed in the next 15 years to absorb the growing global workforce, mainly in Asia and Sub-Saharan Africa".

The aim of this work is to investigate the feasibility of agile methodology to manage the development of in-house medium-sized projects for SMEs using two database technologies, the RDB and ORDB. Such methodologies enable us to sustain IT communities in terms of cultural behavior, group coordination, and resources in order to develop our institutions, social capitals, states and eventually the region. To fulfill this aim, two applications for the Student Registration System for a university located in Lebanon are developed using traditional and agile methodologies. Further, RDB and ORDB technologies are used in the implementation. Results show that agile method overcomes traditional methods by reducing the development, testing, and maintenance time; and thus reduce the cost of IT implementations in SMEs. Further, RDB had better performance over ORDB in retrieving simple data; however, ORDB beat RDB whenever data becomes more complex and manipulated with complex queries.

2. Literature Review

Management methodologies have been extensively researched during the past decades. In researching project management methodologies, Kloppenborg and Opfer (Kloppenborg & Opfer, 2002) performed a profound study that spans the period from 1960 till 1999 covering more than 3,000 thesis, periodicals and reports. From this research we notice that 62% of the research was conducted in the 1990s with 33% of this research is related to information

systems project management, simply because the 1980s and 90s witnessed the advancement of IT industry worldwide. Since then, organizations became more dependent on IT projects, and thus, they experienced the need for methods to tailor the IT project development that reduces cost, time and use resources. Several studies noticed that the majority of the literature on system management methodologies targets large organizations, as these organizations can afford the financial burden to implement such systems (Xu & Quaddus, 2005; Maguire & Koh, 2007). Several methodologies are researched in this regard, the most popular ones include: Prince2, V-Modell, Six Sigma, Waterfall, Spiral model, Iterative approach, and Agile methodologies. A brief description of these methods is discussed below.

Prince2 (**PR**ojects **IN** **C**ontrolled **E**nvironments) was adopted by the British government for to control complicated IT projects (OGC, 2005). It is composed of 45 process divided into 8 groups. Many management specialists see that such a method might be promising for large projects; however, it's questionable for small projects in SMEs (Nicholson, 2004). The V-Modell project management method was introduced by the German government in 1993. The V-Modell method is also rich with processes classified into project phases. By the end of every phase a decision should be taken about moving to the next phase if declared goals were achieved. In fact, the founder of this method state that more efforts are needed as project size becomes smaller (BMI, 2006). Motorola developed the Six Sigma method in the 1990s. Researchers describe the process as "a data-driven structured problem solving methodology for solving chronic issues facing business" (Joshi, 2012). Industry view Six Sigma as a very expensive method to be implemented in small businesses to implement. In fact it performed well in processes of production and planning, however, its bureaucracy and inflexibility may lead to delays and suppress creativity (DeMerceau, 2016). Another popular model which has been used since the 1970s is the waterfall model. This model produces software in phases. The deliverable of each phase is an input for the next one (Royce, 1970); therefore, failing to produce a deliverable in one phase will prohibit the following phase from proceeding; this leads to delay the entire development process. One disadvantage in this model is its inflexibility in terms of handling forthcoming changes raised by customer; as the customer may not see the delivered product until it is almost finished. Such changes will be very expensive to implement especially if they were introduced in later phases (Boehm, A spiral model of software development and enhancement, 1988). Such disadvantages led to the introduction of the spiral model which produces usable prototypes of the end product in the early phases, thus, allowing customers to refine the product during development (Munassar & Govardhan, 2010). Accordingly, the spiral model studied the risk at every iteration and developed the product incrementally (Boehm, A spiral model of software development and enhancement, 1988). Although the spiral model was an improvement for development, it is identified with some weaknesses: it is an expensive model to use and needs qualified users to perform the risk analysis phase which is essential for the success of the project and most important it is not recommended for smaller projects (Munassar & Govardhan, 2010). In the 1990s, a mixture of the waterfall and the spiral model was introduced under the name of iterative model (Larman & Basili, 2003). An improvement to this model lead to the Rational Unified Process (RUP) software development framework (Kruchten, 2004) which still attracts many organizations today because of its flexibility to be

customized to specific projects; further, its phases could run in a sequential or concurrent manner depending of the project specifications (Leffingwell & Widrig, 2003).

Based on the above mentioned models, two popular software development standards emerged, the ISO/IEC 12207:2008 international standard (ISO/IEC-IEEE 2008, 2008), and CMMI (Capability Maturity Model Integration) model (Boehm & Turner, Balancing agility and discipline: a guide for the perplexed, 2003). These standards formed the base for new system development methodologies that targets small software projects developed by small organizations (Laporte1, O'Connor, & Paucar). Such organizations face challenges when implementing traditional software development methods because of the several needs these methods require and the difficulties faced when developing software for unstable environment. Such difficulties and challenges were best faced with the agile methodology for developing software (Šochová, 2009).

Agile methods depend on four principles declared in the "Manifesto for Agile Software Development" (Beck, 2001). These principles address (1) Individuals and interactions over processes and tools, (2) Working software over comprehensive documentation, (3) Customer collaboration over contract negotiation, and (4) Responding to change over following a plan. Such principles force team members to coop with any unexpected requirements or changes. In fact, the main goal of agile methods is to encourage cohesiveness in teamwork enabling SMEs to react more quickly to change and to produce best results faster than ever, overcoming problems faced in traditional methods. Several versions of agile-development were proposed, but the most popular ones are Scrum and Extreme Programming methods. Scrum method is iterative, incremental process that is predicated on a team-based approach. It relies heavily on the three concepts of visibility, inspection, and adaptation. As for visibility, it should be visible whenever successful completion of tasks is established. The inspection concept requires inspecting the output of each team at regular intervals, to check whether there is any deviation from the proposed goal. In adaptation, the appropriate development methodology is adapted whenever a deviation is discovered. In this work, we adopt the Scrum method for it is known to be flexible, adaptable, empirical, productive, and iterative method (Sanchez & Nagi, 2001).

Several researchers studied the implementation of complex IT projects in terms of risk management (Glass, 2006; Cooper, Grey, Raymond, & Walker, 2005) or improving software development for large projects (El Emam, 2008); however, only few studies targeted IT project creation and integration in SME environment. The lack of such studies has been identified by several researchers (Fioravanti, 2006; Augustine, Payne, , Sencindiver , & Woodcook, 2005; Benko & McFarlan,, 2003) who encouraged to investigate successful development methods for quality software constrained with time limits in SMEs, especially in dynamic and unpredictable situations. Our work aims to contribute in filling this gap and to produce a work that future researchers to build upon.

3. Theoretical Framework

One of the main problems that system developers face is adapting to changes (Austin & Devin , 2009). Sometimes, system requirements change after a project starts, and customers often change their expectations for the final software or system. Traditional development life cycle methods usually facilitate responding to and accommodating changing requirements, but these facilitations could be costly and time consuming. Accordingly, many system developers adopt methods that allow better and improved agility.

This research draws on Information processing theory (Galbraith, 1973; Tushman & Nadler., 1978). The theory postulates that an organization should seek to achieve a fit between its information processing requirements and needs and its information processing abilities or competencies. The characteristics of and the interdependencies among core business processes define information processing needs. An organization's competencies in information processing originate from its ICT resources, strategies, structure, and levels of informal coordination. If the information processing needs of an organization are low, then planned and standardized coordination will be supported. However, high information processing needs may trigger informal communication and collaboration. A mismatch between an organization's information processing needs on one hand and its coordination/communication structure may result in wasted resources as well as poor information processing performance (Galbraith, 1973; Tushman & Nadler., 1978).

Another factor to take into consideration is the nature of project interdependencies, which could be changeable and unpredictable sometimes (Thompson, 1967). Thus, changes in certainty levels within an organization's operations, strategies, and environment would lead to changes in interdependencies. Uncertainty and risk levels increase if the developers lack experience, knowledge, or team working and coordination skills. This risk poses a greater threat in larger and more complex working environments.

Organizations can minimize risk by optimizing the formal structure, enhancing flexibility, or selecting communication methods that are more appropriate to the task (Dennis, 2008). For example, if reciprocal interdependencies are high, managers should formally group those roles and individuals to reduce the cost of their coordination. Managers can also define the software modules development process in a way that minimizes reciprocal interdependencies. Finally, if managers properly group roles and define tasks pertinent to each module, they can adopt and implement the appropriate information technology that would support reciprocal and sequential coordination among groups. These strategies, if properly executed, can help create better alternatives for software development methodologies and supporting coordination and communication structures.

4. Research Questions

The objective of our research is to improve the current situation in the organization with the help of individuals working in teams to address issues and solve problems within that organization. Further, this research is conducted in an organization that is completely restructuring and changing its business processes; thus we need an approach that has a degree of flexibility to respond to frequent issues arising in the organization. Accordingly, this research is based on the action research method, and attempts to answer the following research question:

Research Question 1: To what extent can the scrum agile methodology overcome traditional methodologies in terms of time and cost while managing IT implementations in SMEs?

Getting back to the literature review, it is hypothesized that agile methods can produce promising results in the management of IT projects; however, this work is more than a theoretical research, thus, an application for SMEs is developed to test the effectiveness of the proposed method in the above research question. Developing such application led us to the following two research questions:

Research Question 2: what database technology best fulfills the requirements of the development methodologies mentioned in research question 1?

Research Question 3: which database technology gives better results on data sets generated in SMEs?

The last two research questions will help us understand whether database technologies used in implementing IT projects would affect the methodology used for implementation and which database technology would best fit SMEs data sets.

5. The Scrum Methodology

The Scrum model of the agile methodology manages the system development process by addressing three basic components: Roles, Process, and Artifacts. The roles are played by Product Owner, Scrum Master, and the Scrum Team. The process is classified into the following major activities: the Sprint Planning Meeting, the Sprint, the Daily Scrum, and the Sprint Review Meeting. Scrum Artifacts include the Product Backlog, the Sprint Backlog, and Burn down Charts. Below we describe each of these components.

Roles:

The roles component clearly describes the role of each stakeholders involved in the development process. The Product Owner is, usually, an operational manager who knows about the project and the progress of builds. The Scrum Master manages the team in order to fulfill the required tasks and achieve its goals. The Scrum Team typically is composed from five to ten members dedicated to work on the project. The leadership of the team will rotate among team members per iteration (sprint).

Process:

The first activity of the process is the sprint panning meeting. It is held at the beginning of each sprint (iteration). Participants of this meeting are the product owner, the scrum master and the scrum team. This meeting is done in two phases and may span up to a full day. In the first phase, the product backlog and the sprint goal are determined. In the second phase, the sprint backlog is created. The second activity of the process, the sprint, can start now since the first activity is fulfilled. Sprints are limited to a four weeks iteration cycle. During a sprint execution, interference with the scrum team is prohibited; and thus project requirements cannot be changed during a sprint. The third activity is the daily scrum. It is recommended to start each sprint with a 15-minutes daily scrum meeting between the scrum master and the scrum team where every team member briefly talks about what he did since the last scrum, what he is going to do till next scrum and what blocked him from getting his work done. The main goal of the daily scrum meeting is to track the progress of the team and to allow team members to make commitments to each other. The last activity in the process component is the sprint review meeting which is held at the end of each sprint. In this meeting team members presents, informally, to the product owner the functionalities created during the sprint.

Artifacts:

The product backlog is the first artifact created by members of the sprint planning meeting. It is an ordered list of backlog items that represent the project requirements stored according to a given priority. It consists of a list of features that should be included in the end product. During the sprint planning meeting, the team discuss the story points required to fully implement product features and thus performs an estimation of each product backlog item to place it in a size category, and to estimate the amount of hours needed to complete that item. Such an activity will give an idea on the effort needed to handle that item within a sprint. The second artifact is the sprint backlog. A sprint backlog is created for a particular sprint. It is a subset of product backlog which is created only by scrum team members who can add or subtract items from the sprint without exceeding 300 tasks per sprint. The team may need to break down a task if it is determined that it will take more than 16 working hours. The third artifact is the burn down charts which are used to monitor the work done. It is a graphical representation that shows the time spent to finish the required tasks. This chart will decrease with time when members of the team finish their tasks. Burn down charts are used to track the progress of a sprint, a release and the overall project progress.

6. Database Technologies

In today's information age, databases became a necessary and a very essential part of information systems where features like reliability, recovery, robustness, and data integrity is needed. Several database technologies exists today, two of the most popular ones are relational databases (RDB) and object-relational databases (ORDB). According to Pardede (Pardede, Rahayu, & Taniar, 2005), both RDB and ORDB provide the above mentioned

feathers; however, ORDB extended RDB using SQL:2003 standard. Further, it supports object oriented notions such as inheritance, encapsulation, polymorphism, and user defined types (Pardede, Rahayu, & Taniar, 2005; Rahayu, Chang, & Taniar, 2001). The following subsections highlight on the important features of these two database technologies.

5.1 Relational Database Concepts

The relational database model, which was first introduced by Codd (Codd, 1970), have pioneered data management in software production since the late 1970's (Leavitt, 2000). It stores data in non-redundant tables called relations. There exist three basic components of the RDB: data structures to store data, integrity constraints, and operations. The data structures include the fields, rows, tables, instances, and the schema. The integrity constrains include the domain, entity integrity (key), and referential integrity (foreign key) constraints. The operations that are provided in RDB allow the user to SELECT, INSEERT, UPDATE, and DELETE data from the relation. Beside these operations, access control to grant or revoke access to database objects is implemented using a language called SQL. Further, RDBs are known of data independence between the application and the database.

Relational database systems are known of their maturity and simplicity (Pardede, Rahayu, & Taniar, 2005). However, the RDB does not support complex data types such unstructured data, and inheritance relationships (Silberschatz, Korth, & Sudarhan, 2006); further, the join operation in relational databases reduce the query processing efficiency (Dietrich & Urban, 2005) especially when transforming an object oriented inheritance relationship into relational tables.

5.2 Object Relational Database Concepts

Based on object-oriented databases (OODB), the object-relational database (ORDB) technology was first introduced in the middle of 1990s. In fact, an ORDB management system supports relational databases and object-oriented concepts (Thomas & Begg, 2001). The ORDB core data model represents objects as relational tables. Thus designers can use SQL-based data definition language to represent object oriented concepts in ORDB (Krishnamurthy, 1999). The ORDB concepts, like abstraction, encapsulation, inheritance and polymorphism, are inherited from the Object Oriented concepts. The basic components of the ORDB are Object Types, Methods, and Nested Tables. Object Types, these are user-defined data types (UDT) or abstract data types (ADT). Methods in ORDB are encapsulated in their corresponding objects; these are the behavior of the object defined by the user to access object data. Nested tables are collection types that can be stored within another table. Today, since RDB is unable to represent object oriented programming languages features and because the need to store complex data from digital media had increased, considering ORDB became a necessity.

In ORDB, when objects are created, the DBMS engine assigns a unique reference to that object called 'OID'. It is worth noting that an OID cannot be modified or reused, and thus, OID allow objects to be unique. Further, objects reference each other using the OIDs. In addition to the OIDs, primary keys could still be used as in RDB.

One of the advantages of ORDB over RDB is the elimination of the 'impedance mismatch' which is the "incompatibilities that occur at each interface between two set of tools due to the different models for importation representation" (Egenhofer & Frank, 1992); thus impedance mismatch between application programs and DBMSs not only affects the time to develop and application but also the application performance (Egenhofer & Frank, 1992). Still, it is worth mentioning that development time and application performance resulting from transforming objects to relational tables depends on the complexity level of objects used and the nature of data to be stored (Ambler, 1998). For simple application objects, the developer can easily map these objects to relational tables with negligible development time.

7. Case Study

In order to validate our proposed approach for developing information systems in SMEs, a case study is conducted in a small university located in Beirut, UniversityB[1]. UniversityB is a private university in Lebanon established in 1996 and comprises 70 employees now. It aims at giving special attention to scientific research and development; further it provides a means of higher academic education in different domains.

The information system that was a need for UniversityB is a *Registration System* that tracks the students' activities and allows them to register their courses and generate the needed reports and attestations. The appropriate settings required for implementing the case study are determined; all needed hardware, software tools, and database technologies made available at the project site. In software development several components are involved. In this case study the Scrum methodology is implemented and the corresponding components are identified as follows. Actors in this project were clearly identified as the *product owner*, the *team* and the *scrum master*. The role for each actor is assigned; it is described together with the management process below.

6.1 Actors and Roles

Product Owner: the product owner is the representative of the Student Affairs Department at UniveristyB. He sets the direction of the project and knows what functionalities and tasks to be built by the team and in which sequence these builds should progress. The list of these functionalities and tasks forms the *product backlog* which is described in Section 6.2.

The Team: the team is responsible for implementing the *product backlog* which was created by the product owner. In this study the team is composed of six members who are self-managing and responsible for finding how to best turn the product backlog into increments within iteration (sprint). The members of the team are classified into two developers (Developer 1 and Developer 2), one database designer and administrator (DB designer), two system analysts and designers (System Analyst 1 and System Analyst 2), and one testing specialist (System Tester). Developer 1 joined UniversityB in June 2008, Developer 2 and DB designer joined UniversityB in October 2010. System Analyst 1 and System Analyst 2

[1] In this work we use pseudonyms to guarantee the privacy of the university and participants in this case study.

have been working at UniveristyB since September 2007. The System Tester was newly hired in September 2014.

The Scrum Master: In our approach, the role of the scrum master is only related to the scrum process which is completely different from the role of the project leader. His is responsible for helping the product owner and the team to understand and apply Scrum to the project. Thus he needs to coop wisely with the company culture and adapt the appropriate method to handle obstacles so that the team can concentrate on the development process. The Scrum master joined UniversityB in September 2007.

6.2 The Product

The product to be developed was a Student Registration System. The system should provide students with all facilities to register their course and track their academic progress towards graduation. It generates all needed reports, transcripts, and statement needed by the students and the administration. Further, the system keeps a record of students' payments, and generates the needed reports to control the due dates to settle differed payments. Some of the problems UniversityB faced in the old system are that it was hard to follow up and check students' progress towards graduation. Another problem is that students register courses without taking prerequisites. The registration process used to take a long time because of the bureaucratic and semi-automated procedures followed to complete the registration process. Student services, such as generating statement of enrolment or student financial information, followed a hectic procedure and took a long time to be completed. The proposed solution consists of a user friendly online system with web and mobile interfaces and server software for storing, managing and connecting various databases to look up needed information and generate the required reporting. This product is developed in two versions, one using the RDBMS and another using ORDMS.

6.3 Project Startup

On the 20th of October 2014, the project was lunched. The first meeting was held between the *scrum master* and *product owner* to prepare the product backlog, the meeting was adjourned after scheduling a full day meeting for *product owner, scrum master* and the *team* to discuss the product backlog. In this meeting an estimate of the needed amount of work to implement product backlog is set, and the product backlog items are prioritized and divided into several sprints. The other days are reserved for the sprint lifecycle.

6.4 The Development Method:

The development method selected is a method with iterations each spans two weeks and builds are delivered once every three months. As mentioned above this method is inspired by the Scrum method. Every iteration (sprint) starts with a planning meeting. The deliverable of this meeting is a clear list of tasks to be completed by the end of the sprint. Team members arrived to the office between 7:30 a.m. and 8:30 a.m. At 10:00 a.m. they start the daily scrum meeting before getting back to work on their tasks. Each of the actors involved in the development process was playing his role precisely as described in Section

6.1. Accordingly, *the product backlogs* and the *sprint backlogs* are now ready, and the sprint has begun.

6.4.1 Product backlog

The product backlog is an ordered list of tasks of different sizes and priorities that the team should work on to complete the product. It is the only source of product requirements. It lists all features, functionalities, enhances, and fixes to be made to the product in later releases. In our case study, the product backlog had on average about 150 tasks. In fact the product backlog is dynamic and never complete; it evolves during the development of the product. Initially, it contains the initial best understood product requirements. Whenever errors or missing features of the product are found, an email is sent to the developers through the product owner. These requests are added to the product backlog by the team and the product owner is then responsible for assigning a priority to the item. A Sample of the product backlog used at UniversityB is shown in figure 1.

Item #	Description	Priority	Estimate	By
1	Develop first draft of requirements for use	58	12	MF
2	Databases and tools installation	58	6	MM
2.1	Database configuration	58	3	MM
2.1	Test Database connectivity	58	1	HM
3	Agree on name and file standards	62	2	AA
4	As a student, I want to register a course	62	52	AA
5	Create Admin GUI	62	18	HM
6	As an advisor, I want to check course req.	55	40	
7	Diagnose and fix transcript generation	65	15	AA
8	Improve performance list of student grad.	55	10	HM
9	Check history of a specific student	50	15	HM

Figure 1. Sample of the Product backlog

6.4.2 Sprint backlog

The sprint backlog is a list of tasks selected from the product backlog during the sprint planning meeting plus a plan for delivering the product increment. It should be completed during the current sprint. The sprint backlog is a subset of the product backlog, but it often changed during the sprint as new tasks were discovered and added. Thus, it is a plan with enough details that changes in progress and can be understood in the Daily Scrum. While working on tasks, the estimated remaining work is updated and a task will be removed from the plan whenever it is done. The sprint backlog is updated by the development team only. A Sample of the sprint backlog is shown in figure 2.

Item Description	Tasks	BY	Day 1	Day 2	Day 3	Day 4
4. Student Registration	Home Page	AA	1	0	0	0
	Login	AA	1	0	0	0
	Search Results page	AA	3	0	0	0
	Create/Modify Tables	HM	2	0	0	0
	Create/Modify Stored Procedures	AA	3	5	4	0
	Create/Modify Data Access Layer	HM	6	2	0	0
	Student Object Creation	HM	0	6	0	0
	Create Student Methods	AA	0	3	3	0
	Create Student Page Methods	AA	0	0	1	8
	Create Student Page GUI	AA	0	0	0	1

Figure 2. Sample of the Sprint backlog

6.4.3 Sprint Planning Meeting

In the sprint planning meeting, high priority tasks were clear to all team members. Team members added further detailed description of these tasks and how to accomplish them. The tasks estimates are revised, duplicates or obsolete tasks are also removed. Any ambiguity of the tasks was raised during the meeting by the team members to clarify it. Team members showed high cooperation in choosing their tasks and distributing the work of the coming sprint with the amount of work left from the sprint among them. At the end of the meeting task-assignment and documentation, also updates of backlogs were smoothly assigned to team members.

6.4.4 Sprints and Daily Scrum

The sprint backlog list was the source of tasks to be worked on in every sprint. Each completed task status is updated in the sprint backlog from development to testing. After testing is completed and approved it is moved to the list of completed items. Sprints started daily with a daily scrum meeting as described under process subsection in Section 4.

6.4.5 Sprint Review Meeting

At the end of the sprint, the team held a sprint review meeting with the product owner. In this meeting, the current and goals of next sprint are discussed and thorough planning is done after a detailed discussion with the product owner. Any fixes required from the product owner is set as their goal; further, any of the team members can come up with tasks they wish to include in the sprint and vote to whether to include them or not. Further, current practices are discussed and evaluated to decide whether to keep or modify them. The Burn down chart is also updated accordingly. The results of this meeting are carried to the next sprint planning meeting.

6.4.6 Project Completion

As time moved on, sprints were completed; accordingly the product owner received increments of the product. Some of these increments were revised others were accepted. The complete project was deployed in July 2015. The project is a live unit now in UniversityB with refinements and maintenance is still applied on it to meet with the new requirements of administrative board.

6.5 Application Versioning - Database implementations

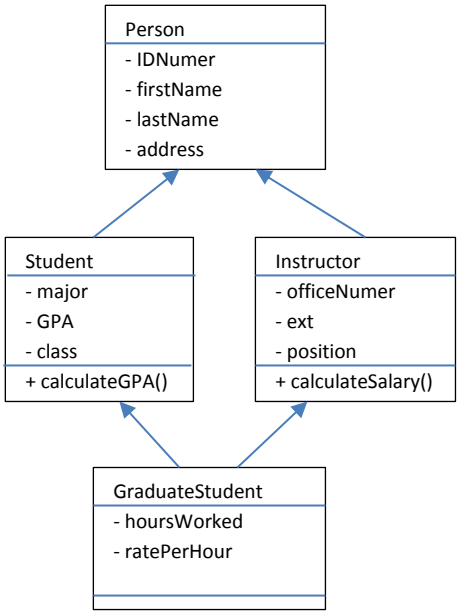
The aim of this work is to study not only the methodology of development, but also to understand whether the selection of the underlying technology used for implementation would affect the development process in terms of time and cost and whether the performance of the application in retrieving data and manipulating its information is also affected. Thus, two versions of the application were developed. The first version is based on the relational databases (RDB) and the second one is based on the object-relational database.

A Student registration system, like any information system is composed of real-world objects (entities) and relationships among them. These real-world objects and relationships were represented in an entity-relationship (ER) diagram using modeling notations like Unified Modeling Language (UML) (Harrington, 2000); Additionally, they were represented as a collection of objects whose behavior, state, and relationships are manipulated using object methods, stored procedures, and SQL query language. Object behaviors are implemented in methods encapsulated in objects. For testing purposes, alternatively, some behaviors used to retrieve and manipulate objects are implemented in stored procedures developed using PL/SQL. The selection of the modeling notation is very important since it allows database designers to appropriately create database tables that stores database objects and relationships.

In referring to the sprint backlogs that keep records of RDB and ORDB implementations, we noticed that the time, and eventually the cost, of developing objects in ORDB is better or, in some cases, similar to the time taken in RDB implementation. That depends on the object types whether it is complex types, user-defined or simple objects. If the application object is easily mapped into a relational table, then the development time is almost the same as ORDB; however, as the complexity of the application object increases, the development time and thus the cost of development increased. This experiment completely reflects Thomas argument (Thomas & Begg, 2001) that using ORDB offers the software designers to think in a natural way at a higher level of abstraction to represent real-world objects and relationships; which eventually leads to reduce the design and development of software applications.

Further, the time needed to develop the ORDB-based application is less than the time taken while developing the RDB-based application. This is simply illustrated in the following example. Consider the mapping of Person object inheritance hierarchy into relational tables shown in figure 3.

To create the relational tables, four CREATE SQL statements are needed, and to populate these tables we need to write four INSERT SQL statements for each of the four tables that will execute eight times as per figure 3. On the other hand, we need to write only three INSERT SQL statements, one for each subclass that will execute only three times. Figure 4 shows a sample SQL code for inserting Students and Instructor types.



Person

IDNumber	firstName	lastName	address
1234	Jad	Barakat	Beirut
1235	Walid	Beiruty	Beirut
1236	Ziad	Bekaii	Beirut

Student

IDNumber	Major	GPA	class
1234	ITM	3.2	Sp
1236	ITM	2.7	Jr

Instructor

ID	officeNumber	Ext.	position
1235	SH 3014	1437	Professor
1236	BB 1301	1923	GradAssist

GraduateAssistant

IDNumber	hoursWorked	ratePerHour
1236	10	12

Figure 3. Mapping object inheritance hierarchy into relational tables

```
SQL> insert into PersonTable values (StudentType('1234', NAME_TYPE('Jad','Barakat'), AddressType('Beirut'), 'ITM', 3.2, 'Sp') );
1 row inserted

SQL>insert into PersonTable  values ( InstructorType('1235', NameType('Walid','Beiruty'), AddressType('Beirut'), 'SH 3104', 1437,
'Profesor');
1 row inserted
```

Figure 4. SQL statements that inserts values into student and instructor tables in ORDBMS

As for the cost of retrieving information, it is expected that ORDB based application will perform better than the RDB based application. The reason is again illustrated in the example shown in figure 3. In RDB implementation, each of the objects in the object-oriented hierarchy is mapped into a relational table. Thus, to retrieve information from more than one table, the RDBMS must join the tables involved in the query. In fact join operation in RDB is very expensive (Dietrich & Urban, 2005); on the other hand, the ORDB uses the OID to refer to objects which enhances the performance because it does not use the join operation to access data in related tables (Harrington, 2000). To illustrate with an example, consider figure 5 that uses object references in ORDB.

```
SQL >CREATE TABLE CourseTable(
                courseID varchar2(25),
                courseName varchar2(25),
                courseDesc varchar2(100),
                inst REF instructorType scope is instructorTable);


SQL > SELECT  c.* ,c.inst.firstName, c.inst.ext
      FROM    CourseTable   c
```

Figure 5. Object references in ORDB

In figure 5, the first SQL statement shows how course table use the object reference ID to refer to Instructor table. The second SQL statement shows how data is retrieved from both objects using the object reference with no need to join tables as done in RDB.

6.5.1 RDB and ORDB performance analysis

The performance testing for RDB and ORDB applications was performed on Intel Pentium 2.5GHz with 4GB of memory. Test cases are written for SQL insert, select, update, and join statements. The computer is restarted after each test to avoid any cashing of data. Each of these statements is applied on several tables which differ in the number of attributes; this will give us an idea whether table size affects the performance of the database technologies under study. Three test sequences are written for each test case. The size of the test sequences is benchmarked 10, 100 and 1000 rows (Lee, Kim, & Kim, 2000). After applying these test sequences, it was noticed that for inset, select, update and delete statements the time of execution of RDB is similar, even better, than that of the ORDB especially for small-sized tables. ORDB had execution time enhancement for large data sets. Further, ORDB had a better execution time when using the object reference ID to access related tables instead of using join operation. Details of these results are found in the below sections.

6.5.1.1 Insert Statement

Test sequences used to test the insert statement are of sizes 10, 100 and 1000 rows. Figure 6 shows the results for size 1000 rows. Results for inserting rows into relational tables and the corresponding object tables with one type and small table sizes ranging from three to five attributes shows performance of three percent faster for the RDB over ORDB. However, as table sizes increased till eight attributes and objects had complex types, the performance of ORDB was faster than the RDB in 8 percent.

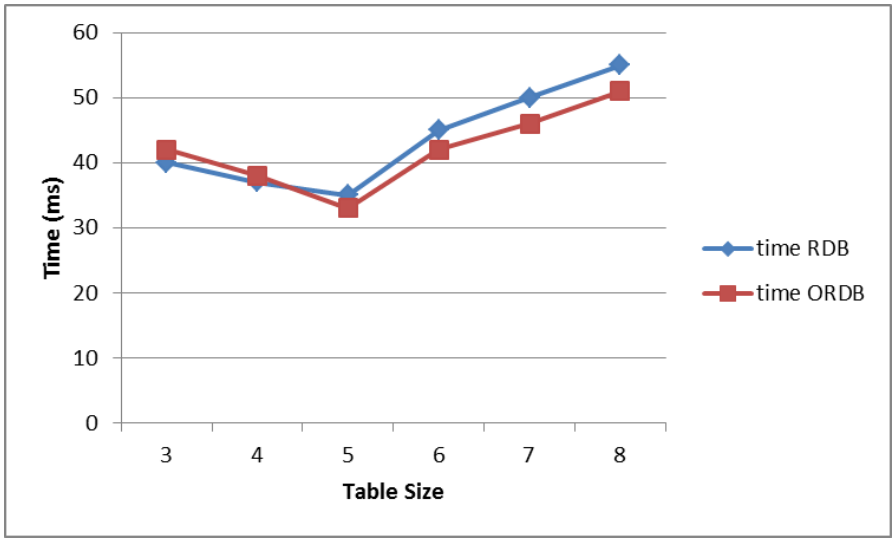


Figure 6. Insert statement execution time – RDB vs ORDB

This could be justified as follows, for objects with simple types and small sizes; there is no need to use the OID to refer to other objects. Memory used is enough to hold large data. In this case the ORDB cannot use the OID as an advantage over the RDB, and therefore, technically both RDB and ORDB have similar underlying structure, and thus the execution time is almost the same. On the contrary, when inserting into tables with more complex types and large data sizes, the advantage of using OID here made a difference of eight percent faster execution for ORDB.

6.5.1.2 Select Statement

Similar test sequence sizes are used to test the select statement. Figure 7 shows the results for size of 1000 rows selected. Results for selecting rows from relational tables and the corresponding object tables with one type and different table sizes ranging from three to eight attributes shows performance of four percent faster for the RDB over ORDB.

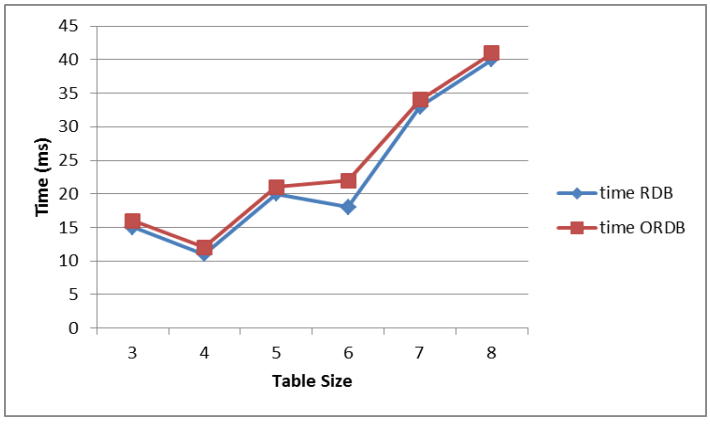


Figure 7. Select Statement execution time – RDB vs ORDB

6.5.1.3 Update Statement

Again, similar insert and select statements same test sequence sizes are used to test the update statement. Figure 8 shows the results for size of 1000 rows updated. Results for

updating rows from relational tables and the corresponding object tables with one type and different table sizes ranging from three to eight attributes shows performance of one percent faster for the RDB over ORDB.

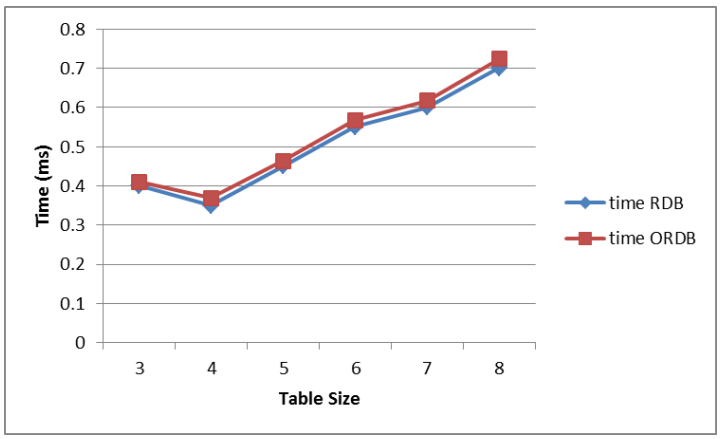


Figure 8. Update statement execution time – RDB vs. ORDB

6.5.1.4 Relational Join and Object References

To test the relational join and object reference performance in retrieving data from different tables, we applied the SQL code shown in figure 9 on Instructor and course tables. For testing purposes, the Instructor table is populated with 900 rows and Course table is populated with 2500 rows.

```
SQL>select  *
        From CourseTable c, InstructorTable i
        Where  c.IDNumber=i.IDNumber;
```

(a) Relational query

```
SQL>select   c.courseID,
             c.courseName,
             c.courseDesc,
             deref(c.instructor)
             From CourseTable c;
```

(b) object relational query

Figure 9. Relational and Object Relational queries to retrieve data from two tables.

The execution time for testing the queries in figure 9 showed that object relational query is forty-eight percent faster than relational query using joined tables. It is worth noting further that as the number of dereferenced objects increased to two objects there was a decrease in the performance to thirty-two percent in order to access the referenced tables. Figure 10 shows the results of this execution.

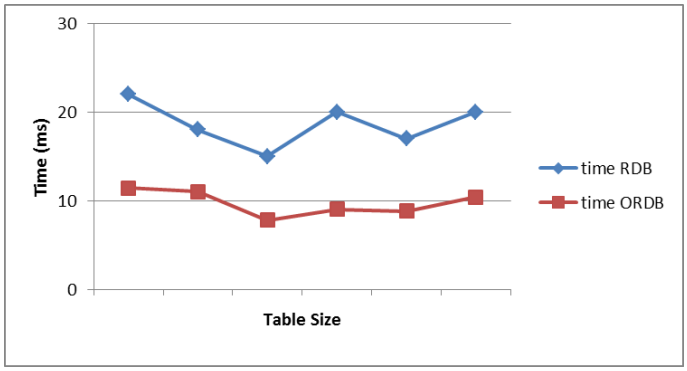


Figure 10. Joined tables Relational query execution vs. object referenced query execution

8. **Discussion and Conclusion**

We presented an agile methodology that has the potential to promote sustainable development through helping SMEs in reducing the time and cost of IT project development. The results came to demonstrate one of the "Agile Principles" that agile processes promote sustainable development (Beck, 2001; Kumar & & Bhatia, 2012). The Scrum methodology presented was able to reduce the gap between IT and business people, leading to generate a successful and efficient end product. A case study was conducted. Results of this case study show that this methodology reduced the cost of development by thirty percent and the level of cooperation between IT developers and product owners is leveraged. Further, this study highlighted on the selection of database technology to implement such projects. Two versions of the application were developed; one using relational databases and the other using object-relational databases. Based on our results, we came to a conclusion that the selection of the database technology depends on the complexity and interrelationships between the data used. Relational databases would be recommended for simple data with simple queries; however, ORDB should be selected as the data becomes more complex and manipulated with complex queries. Such contingency approach – applying the right development methodology taking into consideration the volume of data and its complexity – can help a company realize higher efficiency levels in time and cost, thus contributing to sustainable application development and use.

Moreover, we aimed in this work at researching the best practices to be followed by SMEs while developing in-house projects, and whether the database technology implemented has an effect on the sustainable development of software in terms of time and performance. Accordingly, our goal was not to come up with new project management methodology; however, the goal is to customize existing methodologies to help SMEs produce projects with the minimal cost, and select the appropriate database technology that fulfills the needs of the developed applications. The results of our work show that the Scrum methodology is able to manage such projects successfully, where the project environment is frequently changing and the project requirements are not completely defined. Although the Scrum methodology was not implemented as it is described in books, it drew a clear guideline on how to manage such projects. Scrum flexibility allowed the Scrum team to manage their project effectively.

In comparison with previous projects, that eventually failed at UniversityB, the Product Owner reported that this project reduced the development process cost by thirty percent; what is even better, is that the stakeholders reported that the final product "ended up to be what they really wanted". Further, the product owner was very satisfied about the cooperation that existed between IT developers and business people, who see the project from different points of views. This pushed towards healthy discussions that led to compromising good solutions. Further, the developers reported that this development methodology gave them an added value to their previous experience. We believe that the above discussion answered the first research question which asked about the '*extent the scrum agile methodology can overcome traditional methodologies in terms of time and cost while managing IT implementations in SMEs'.* In other words, the scrum agile methodology proved to contribute to sustainable application development better than traditional methodologies.

As for the second research question, which asked about the '*database technology that best fulfill the requirements of the Scrum development methodology',* it was clear from the sprint backlog and the list of completed tasks that the abstraction of the ORDB modeling enabled the development team to represent the real-world data in their applications in a faster and simpler way than the RDB models were able to do. This reduced the time to model the data in the database and made the communication easier between development team and the product owner who became more involved even at the level of data modeling. So, in terms of sustainable customer experience and satisfaction, the ORDB is a better enabler than RDB.

The third research question, which asked '*about the database technology that gives better results on data sets generated in SMEs'*, is a very important question since it helps SME application developers to make a decision about what technology could be used to enhance performance. The results of our work show that there was minor difference in performance between RDB and ORDB whenever we are dealing with simple data and simple queries, such as select, insert and update which is related to single tables; however, ORDB performed forty-eight percent better than RDB when objects had complex data types and querying is related to joined tables. Further, it is worth mentioning that this performance degraded as the number of references from one object to other objects increased. This leads to the following conclusion that SMEs can select the database technology by studying the frequency of access to joined tables and the kind of queries needed. Further, they need to understand the complexity of data represented in their databases. For simpler data, RDB would have a noticeable performance over ORDB; For more complex data, ORDB should be selected. With such approach, a more sustainable application development methodology could be ensured for SMEs.

References:

Alshayeb, M., & Li, W. (2005). An empirical study of system design instability metric and design evolution in an agile software process. *Journal of Systems and Software, 74*(3), 269-274.

Ambler, S. (1998). *Building Object Applications that Work.* New York: Cambridge University Press and Sigs Books.

Augustine, S., Payne, , B., Sencindiver , F., & Woodcook, S. (2005). Agile project management steering from the edges. *Communications of the ACM, 48*(12), 85-9.

Austin, R., & Devin , L. (2009). Weighing the Benefits and Costs of Flexibility in Making Software: Toward a Contingency Theory of the Determinants of Development Process Design. *Information Systems Research, 20*(3), 462-477.

Babu, S. (2005). *Scope creep is not only inevitable; it's natural*. Retrieved from The PROJECT PERFECT White Paper Collection: uca.eis.googlepages.com/ScopeCreep.pdf

Beck, K. B. (2001). *Manifesto for agile software development*. Retrieved from viewed 11 Dec. 2010, <http://agilemanifesto.org/>.: http://agilemanifesto.org/

Benko, C., & McFarlan,, F. (2003). *Connecting the dots: aligning projects with objectives in unpredictable times.* Boston, Massachusetts: Harvard Business School Press.

BMI. (2006). *V-Modell® XT.* Berlin.: Bundesministerium des Innern.

Boehm, B. (1988). A spiral model of software development and enhancement. *Computer, 21*(5), 61-72.

Boehm, B. (1988). A spiral model of software development and enhancement. *Computer, 21*(5), 61-72.

Boehm, B., & Turner, R. (2003). *Balancing agility and discipline: a guide for the perplexed.* Boston: Addison-Wesley Professional.

Chin, G. (2003). *Agile project management: how to succeed in the face of changing project requirements.* New York: AMACOM.

Cockburn, A., & Highsmith, J. (2001). Agile software development 2: the people factor. *IEEE Computer*.

Codd, E. (1970). A relational model of data for large shared data banks. *Comunications ACM, 13*(6), 377-387.

Cooper, D., Grey, S., Raymond, G., & Walker, P. (2005). *Project risk management guidelines:managing risk in large projects and complex procurements,.* Ltd, Chichester, England.: John Wiley & Sons.

DeMerceau, J. (2016). Advantages & Disadvantages of Six Sigma. *Small Businesses*.

Dennis, A. R. (2008). Media, Tasks, and Communication Processes: A Theory of Media Synchronicity. *MIS Quarterly, 32*(3), 575-600.

Dietrich, S., & Urban, S. (2005). *An Advanced Course in Database Systems Beyond Relational Databases.* Pearson Prentice Hall.

Dogerlioglu, O. (2012, April). Outsourcing versus in-house: A modular organization perspective. *The Journal of International Management Studies, 7*(1), 22-30.

Economist, T. (2009, August 24). *Computing climate change: How much carbon dioxide do computers emit?* . Retrieved from The Economist : http://www.economist.com/node/14297036

Egenhofer, M., & Frank, A. (1992). Object oriented modeling for GIS. *URISA journal, 4*(2), 3-19.

El Emam, K. &. (2008). A replicated survey of IT software project failures. *IEEE Software, 25*(5), 84-90.

EREL, B. R. (2014). *A Lot of IT Projects Fail, But Why?* Retrieved from Software As Services: http://saasaddict.walkme.com/lot-project-fails/

Eskelin, A. (2001). *Technology Acquisition: Buying the Future of Your Business.* Addison-Wesley.

Fioravanti, F. (2006). *Skills for managing rapidly changing IT projects, .* London: Idea Group Inc.

Galbraith, J. (1973). *Designing Complex Organizations.* MA: Addison-Wesley.

Glass, R. 2. (2006). Looking into the challenges of complex IT projects. *Communications of the ACM, 49*(11), 15-7.

Haider, S. A., Samdani, G., Ali, M., & Kamran, M. (2016, May). A Comparative Analysis of In-house and Outsourced Development in Software Industry. *International Journal of Computer Applications, 141*(3).

Harrington, J. (2000). *Object-Oriented Database Design Clearly Explained.* San Diego, CA USA: Morgan Kaufmann.

Hunter, M. (2004). Information systems & small business: research issues. *Journal of Global Information Management, 12*(4), 1-5.

Ilincuta, A., & Jergeas, G. (2003). A practical approach to managing multiple small projects. *AACE International Transactions*.

ISO/IEC-IEEE 2008, '. (2008). *ISO/IEC 12207:2008: Systems and software engineering - software life cycle processes.* Geneva: International Organization for Standardization/Internationa lElectrotechnical Commission & Institute of Electrical and Electronics Engineers.

Jin-Hai, L., Anderson, A., & Harrison, R. (2003). The evolution of agile manufacturing. *Business Process Management Journal, 9*(2), 170-189.

Joshi, A. (2012). Six Sigma implementation using DMAIC approach. *International Journal of Computer Applications. , 1*(4), 1-7.

Kim, W., & Lochovsky, F. (1989). *Object-oriented concepts, databases, and applications.* New York, NY, USA: ACM Press.

Kloppenborg, T., & Opfer, W. (2002). The current state of project management research:trends, interpretations, and predictions. *Project Management Journal, 33*(2), 5-18.

Krishnamurthy, B. (1999). Bringing object-relational technology to the mainstream. *Proceedings of the ACM SIGMOD International Conference on Management of Data and Symposium on Principles of Database Systems.* Philadelphia, PA.

Kruchten, P. (2004). *The Rational Unified Process: An Introduction.* Addison-Wesley.

Kumar, G., & & Bhatia, P. K. (2012). Impact of Agile Methodology on Software Development Process. International Journal of Computer Technology and Electronics Engineering . *IJCTEE, 2*(4).

Laporte1, C., O'Connor, R., & Paucar, L. (n.d.). Software Engineering Standards and Guides for Very Small Entities: Implementation in two start-ups. *In the Proceedings of the 10th International Conference on Evolution of Novel Approaches to Software Engineering (ENASE 2015).* Spain.

Larman, C., & Basili, V. (2003). Iterative and Incremental Development: A Brief History. *IEEE Computer, 36*(6), 47-56.

Leavitt, N. (2000). Whatever happened to object-oriented databases? *Computer, 33*(8), 16-19.

Lee, S., Kim, S., & Kim, W. (2000). The bord benchmark for object relational databases. . *The 11th International Conference on Database and Expert Systems Applications* (pp. 6-20). London: Springer-Verlag.

Leffingwell, D., & Widrig, D. ,. (2003). *Managing software requirements: a use case approach, 2nd edn. .* Boston, Massachusetts.: Addison-Wesley.

Lewis, J., Henry, D. G., Kafura, R. S., & Schulman. (1991). An empirical study of the object-oriented paradigm and software reuse. *Proceedings on Object-Oriented Programming Systems, Languages, and Applications* (pp. 184-196). Phoenix, Arizona, United States: ACM Press.

Maguire, S., & Koh, S. (2007). The adoption of e-business and knowledge management in SMEs . *Benchmarking, 14*(1), 37-58.

Meyers, B. C., & Oberndorf, P. (2001). *Managing Software Acquisition: Open Systems and COTS Products. .* Addison-Wesley Professional.

Moody, D., & Walsh, P. (1999). Measuring The Value Of Information:An Asset Valuation Approach. *European Conference on Information Systems (ECIS'99)* (pp. 20-37). Copenhagen: The idea group publisher.

Munassar, N. M., & Govardhan, A. (2010). A comparison between five models of software engineering. *International Journal of Computer Science Issues, 5*, 95-101.

Newton, P. (2015). *Managing Project Scope.* Free Management Ebooks.

Nicholson, B. (2004). Why PRINCE2 isn't widely used in the private sector. In *Advanced Project Techniques, .* Suffolk.

OGC. (2005). *Managing successful projects with PRINCE2,4th edn.* London: The Stationery Office,.

Olsen, K. A. (2011). *In-house programming: an option for small and medium sized niche companies.* Norway: University of Bergen and Molde University Collage.

O'Neill, M. (2010). *Green IT for Sustainable Business Practice: An ISEB Foundation Guide. .* BCS, The Chartered Institute.

Pardede, E., Rahayu, J., & Taniar, D. (2005). *Object-Oriented Oracle.* London: CyberTech Publishing.

PMI. (2004). *A guide to the project management body of knowledge (PMBOK guide).* Newtown Square, Pa: Project Management Institute.

Rahayu, J., Chang, E., & Taniar, D. (2001). Performance evaluation of the object-relatioanl transformation methodology. *Data Knowledge Engineering, 38*(3), 265-300.

Raisinghani, M. S., & Idemudia, E. C. (2015). *Green Information Systems for Sustainability. Handbook of Research on Waste Management Techniques for Sustainability.*

Rowe, S. (2007). *Project management for small projects, Management Concepts.* Vienna.

Royce, W. (1970). Managing the Development of Large Software Systems. *Proceedings of IEEE WESCON*, (pp. 1-9).

Sanchez, L., & Nagi, R. (2001). A review of agile manufacturing systems. *International of Production Research, 39*(16), 3561-600.

Schwaber, K. (2004). *Agile project management with Scrum.* Microsoft Press, Redmond,.

Silberschatz, A., Korth, H., & Sudarhan, S. (2006). *Database System Concepts.* New York: McGraw-Hill.

Silberschatz, A., Korth, H., & Sudarshan, S. (2009). *Database System Concepts.* McGraw-Hill.

Šochová, Z. (2009). Software Development Methodology for Fast Changing Environment. *The 2009 conference on New Trends in Software Methodologies, Tools and Techniques*, (pp. 379-88).

Thomas, M., & Begg, C. a. (2001). *Database Systems: A Practical Approach to Design, Implementation and Management.* Boston: Addison-Wesley Longman Publishing Co.

Thompson, J. (1967). *Organizations in Action.* New York, NY: McGraw-Hill.

Tima, G. O. (2015). *Small and Medium Enterprises (SMEs) Finance .* The World Bank.

Tushman, M., & Nadler., D. (1978). Information Processing as an Integrating Concept in Organizational Design. *The Academy of Management Review, 3*(3), 613-624.

Vazquez-Bustelo, D., & Avella, L. (2006). Agile manufacturing: industrial case studies inSpain. *Technovation, 26*(10), 1147-61.

Wang, M. (2010). Using object-relational database technology to solve problems in database development. *Issues in Information Systems, XI*(1).

Xu, J., & Quaddus, M. 2. (2005). A reality-based guide to KMS diffusion. *The Journal of Management Development, 24*(4), 374-89.