Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. V (2010), No. 1, pp. 20-41

A Software System Development Life Cycle Model for Improved Stakeholders' Communication and Collaboration

S. Cohen, D. Dori, U. de Haan

Shalom Cohen, Uzi de Haan
Technion, Israel Institute of Technology
Haifa, Israel
E-mail: {shalom1,uzid}@tx.technion.ac.il

Dov Dori
1. Technion, Israel Institute of Technology
Haifa, Israel
2. Massachusetts Institute of Technology
Cambridge, MA, USA
E-mail: dori@ie.technion.ac.il

Abstract: Software vendors and entrepreneurs, who try to introduce an innovative software product to a specific organization or an entire market, enter a long and tedious process. During this process, the market and various organizations evaluate the product from different perspectives, such as software robustness, manufacturer reliability, and corporate need for the product. The vendors and entrepreneurs engaged in this process encounter decision crossroads for which no relevant guidance exists in the literature.

The research closely monitored the processes associated with the introduction and assimilation of an innovative off-the-shelf (OTS) software product into five different organizations in different vertical market segments. Observations were carried out to assess organizational and marketing processes and to document and analyze what the software product undergoes before it is accepted for acquisition or full implementation within the organization.

The research outcomes offer a unified, collaborative multi-tier System Development Life Cycle (SDLC) framework and methodology for packaged OTS software products that greatly improves communication and collaboration among the stakeholders. Each tier addresses a different force or stakeholder involved in the software market: vendor, customer, consultants and integrators. All stakeholders refer to the same time-line thus; tasks of various stakeholders are streamlined. Adherence to the unified time-line brings about an increased amount of stakeholder interaction, communication and collaboration.

Newly found tasks that improve communication and collaboration among stakeholders include (1) offering of the OTS software product together with personnel as a bundle, (2) an improvisation-intensive iterative task of weaving potential customers' requirements into the prototype, and (3) a third sale milestone, representing the successful diffusion of the product. The significance of this interdisciplinary research stems from its unique position at a crossroad between software engineering, marketing, and business administration, which has not yet been sufficiently explored or cultivated.

Keywords: collaboration, system development life cycle model, stakeholders.

1 Introduction

Two major trends dominate the software development world today. The first is the shift of organizations from fulfilling their own software requirements in-house to buying it on the market, either as an

off-the-shelf packaged software product or from a company tailoring a specific solution.[1] The second trend is the shift from developing tailor-made software to purchasing packaged software from vendors either in stores or directly from the vendors.[2] Here we assume that acquisition of packaged software is done by an organizational consumer.[1]

When relating to a packaged software product, which may be seen as a system by its own accord, one should make a clear distinction between a software product and an Information System (IS). [3] An IS is made up of a number of software products or modules put together.[1] This research examines an off-the-shelf (OTS) packaged software product as a system[2] which goes through the various stages of System Development Life Cycle (SDLC)[3]. Many software development processes and models use stages outlined in SDLC.[4] We relate to SDLC not only in its traditional "waterfall" sense, but also to other models outlining the stages in software development. Since these models, including the spiral and Rapid Application Development (RAD) are not as broadly known as the "waterfall" model and are less useful for explaining the market effects on software development delineated hereafter [1], we do focus on the waterfall model as a reference.

The lifecycle of an information system includes the various phases that a software product goes through starting with its conception all the way to the stage when it is no longer available for use.[5] The software lifecycle, depicted in Figure 1, typically includes the following phases: requirements, analysis, design, construction (or coding), testing (validation), installation, operation, maintenance, and the less emphasized retirement.[6]

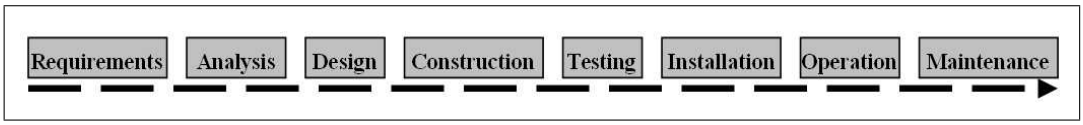

Figure 1: The traditional phases of the System Development Life Cycle Model

These basic phases have also been adopted for IS acquisition purposes. Although the names of the phases were changed where appropriate, the basic structure and timeline have been kept. The phases of IS acquisition, shown in Figure 2, are project justification, financial evaluation of the project, preparations for acquisition, Request For Proposals, vendor evaluation, contract negotiations and, implementation and maintenance.[7]

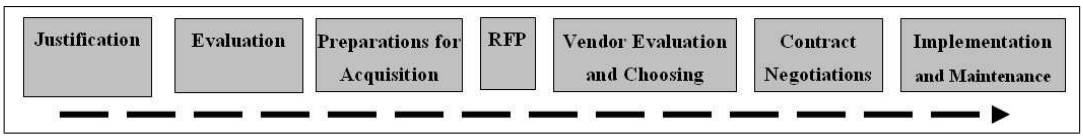

Figure 2: The SDLC model adapted to the acquisition process of Information Systems

The literature research, summarized below, indicates that no significant attempt has been made to extend the SDLC model to other situations encountered by many software vendors and software developing entrepreneurs.[2] Software vendors and entrepreneurs, who try to introduce an innovative software product to a specific organization or an entire market, enter a long and tedious process. During this process, the market and various organizations evaluate the product from different perspectives, such as

[1]The scope of the research is limited to describing the organizational consumer and not the private home user which is a discussion in its own right and differs in many ways from that outlined hereafter.

[2]"The System," "OTS software product," and "Packaged Software Product" will be used from now on interchangeably

[3]The acronym SDLC will denote here the system development lifecycle model as it relates to software products. Its implementation with regards to OTS software products will be discussed here.

software robustness, manufacturer reliability, and corporate need for the product. The vendors and entrepreneurs engaged in this process encounter decision crossroads for which no relevant guidance exists in the literature.

This lack of guidance is somewhat surprising, since information systems development is best understood as a market phenomenon. It is a perspective which highlights how software is developed, who performs the development, who sells the related products, and how they are introduced to users.[1]

2 Research Goal and Objectives

The goal of this research is to develop and evaluate a collaborative multi-tier lifecycle development model for packaged off-the-shelf (OTS) software products. The proposed model accounts for market and organizational factors and the way they are woven into the traditional phases of software development. To this end, the research has monitored, outlined, characterized, defined, and mapped specific phases which OTS software products typically go through. The resulting comprehensive model relates to the development, marketing, assimilation, and other organizational aspects of the OTS software product. The research has identified and defined new, modifiable software lifecycle processes, the adoption of which might benefit various stakeholders under various marketing conditions. Our hope that the prevailing model will entail this aim by creating a task-based learning community which is a group of people who are organized around a task i.e. stakeholders, collaborating for a specified period of time to produce a product.[8]

Here, we attempt to create a unified exhaustive SDLC framework on one timeline with a number of tiers, creating a new collaborative multi-tier system development life cycle methodology. Each tier addresses a different force or stakeholder involved in the software market, such as producers, consumers, consultants, and integrators.[1] The basic time frame of the SDLC, especially the beginning (inception) and end (implementation and maintenance) is kept. The various milestones along the SDLC time line indicate an appropriate task for each tier and an explanation of that task. Tasks on the same vertical axis are to be performed concurrently and collaboratively.

Figure 3 depicts a possible scheme of the proposed collaborative multi-tier market- and organization-oriented SDLC model to be fleshed out as a result of the field study outcomes. The list of stakeholders stacked in Figure 3 is by no means exhaustive.

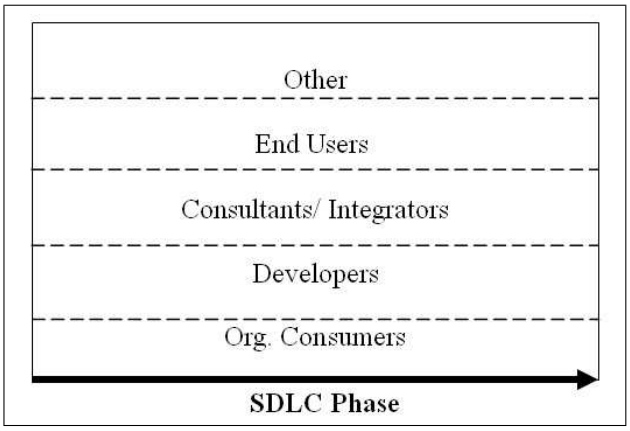

Figure 3: A possible scheme of the proposed collaborative multi-tier market- and organization-oriented SDLC model to be fleshed out as a result of the experiment outcomes

The main novelty of this research is that it is a first field-based study that is aimed at the establishment of a collaborative multi-tier SDLC model and a methodology based upon it. In addition, in most IS field studies, researchers have access to a limited amount of evidence and observations in participating

organizations. In contrast, this research takes advantage of the fact that the software that was examined has been developed at the Technion and was fortunately being positioned at the point of time required for this research. Due to the special ties between the Technion and the software vendor, this study has had access to evidence and observations that are normally out of reach to researchers. The validity of these unique findings was tested against the more abridged findings of the control case studies.

3 Literature Review

The only academic development model incorporating any type of market effects is that of Carmel and Becker.[9] They developed a process model for packaged software development which was partly empirical.

Carmel and Becker [9] point to a few market-related actions necessary to be performed in some, but not all of the described stages. Actions like "assessing product differentiation considerations" are attached to the "Initial Screening Stage" of the "Requirements loop" with no explanation how they can be achieved.

In summary, Carmel and Becker [9] were the first to attempt a complete process model which adds marketing tasks. Their model, however, was only partially based on empirical findings, and instead of having the market define software market needs and SDLC phases, they suggested them a priori. Moreover, as Cusumano et al.[10] noted, their reliance and justification of a pivotal "freeze specification" stage is problematic in a highly volatile market.[10]

As Carmel [2] noted, no major study had been conducted on market introductory effects on packaged software innovations before 1995.[2] Nevertheless, the idea of introducing a market-based perspective into Information Systems development was introduced later on.[1] He juxtaposed a market-oriented approach with a simplification of the traditional "waterfall" model. At the basis of his idea is a separation of the traditional SDLC model (from development to user introduction) into two separate parallel models, one for the software developer and the other for the software consumer. In addition Sawyer had several interesting assertion as to the growing importance of additional stakeholders in the development process.

For example:

- Third parties (consultants, vendor representatives, etc.) have an increasing role in the initial stages of SDLC. Consultants/integrators are now also part of the Information System Development (ISD) process, as they enable and mediate the software market. This contributes to widening the chasm between users and vendors. This chasm is bridged only by indirect links between customers and developers via intermediaries or customer surrogates.[11]

- System installation requires a third party in charge of installing the product, customization, and training.

- The development process is of smaller importance to the consumer than the final product.

Although most of the assertions in Sawyer's model may make sense, they are in no way based on empirical evidence and do not have a direct connection to an SDLC model currently in use. His model lacks due reference to the producer's side, an aspect which this research has elaborated on.

In order to cover a large number of organizational and market-related SDLC influencing factors, we searched for academic and professional models in seven domains. We began by looking at the above mentioned few existing market based IS/Software Development models to see how an innovative OTS software product is produced in the market. We then continued by looking at works on software cycles and structured development studies to uncover the new OTS software manufacturing methods. Leaving the IS domain, we followed Moore's technology diffusion theory to look for models on technology adoption, innovation introduction and marketing diffusion theory which may relevantly describe the

diffusion process of an OTS software product too.[12] We then reviewed research in the cross domain of organizational decision making on IS/Software related issues to learn how the common IS/Software related decision-making processes in various organizations are performed. The maturity of the software product, as well as that of the organization, is a matter of much interest to Industrial Engineers and Business Administrators and it influences entrepreneurial vendors tremendously and therefore reviewed here too. The relatively young academic field of Entrepreneurship was searched for adequate models and research on innovation, innovation-exploration and entrepreneurship in the software market. Finally, we surveyed the market for best practices and existing methodologies for OTS software development by entrepreneurial vendors. Table 1 summarizes the main studies related to this research topic under the various categories. From the above literature review we learnt about the possible variables and added them to the examined research's model as described in the following section.

Tabel 1. Summary of main studies SDLC and related subjects.

Aspect	**Article**	**Empirical**
Market Based IS/ Software Development	Carmel and Becker[9]	Very partial
	Keil and Carmel[11]	Yes
	Sawyer[1]	No
	Sing and Kotze[34]	Partial
Software Cycle and Structured Development	Cusumano et al.[10]	Yes
	Cusumano[35]	No
	Carmel[2]	Yes
	Clark and Wheelwright[36][37]	Partial
	Wheelwright and Clark[38]	Partial
	Boehm and Bose[39]	Yes
	Fine[40]	No
	Avison and Fitzgerald[4]	No
	Ebert[42]	Yes
Technology/ Innovation Introduction/ Diffusion Theory Marketing[4]	Mustonen-Ollila and Lyytinen[43]	Yes
	Lucas and Spitler[44]	Yes
	Davis[45]	Yes
	Moore and Benbasat[46]	Partial
	Brancheau and Wetherbe[47]	Yes
	Cooper and Zmud[48]	Yes
	Fichman and Kemerer[25]	Yes
IS related Decision-Making and Software Acquisition Processes	Verville and Halingten[49]	Yes
	Nelson et al.[50]	Yes
	Iivari and Ervasti[51]	Yes
Software Product and Organization Maturity	Paulk[52]	No
	Nordman [53]	Report
	Lee and O'Connor[54]	No
	Montaguti et al.[55]	No
Hi Tech Entrepreneurship	Shane[56]	Partial
	Shane[57]	No
	Murray and Tripsas[58]	Yes
	Baker et al.[23]	Yes
	Vera and Crossan[27]	Yes
Best Practices	NIH Matrix[59]	Best Practice
	Agile[61]	Best Practice

[4]For the most part, the writings in this discipline have not distinguished between the more general definition of an IT product and a specific IS/software-like product.

4 Methods and Experiments

In this section, we first provide a short explanation to the research method used (a), and then we describe the case study sites selected (b). Data collection efforts are described in (c) and finally in (d) we cover the preliminary research model.

a. Case Study Methodology

Yin identified three main types of case studies based on the purpose for which they are used [13]:

(1) Explanatory - A case study intending to explain the casual links in real-life interventions that are too complex for survey or experimental strategies.

(2) Descriptive - A case study that emphasizes the formation of hypotheses of cause-effect relationships, where a descriptive theory must cover the depth and scope of the case under study.[14]

(3) Exploratory - A case study in which the fieldwork and data collection may be undertaken prior to definition of the research questions and hypotheses. The framework of the study must be created ahead of time to maximize what can be learned, knowing that time is limited. The selected cases should be easy and should include willing subjects.[14]

The research strategy utilizes a combination of exploratory and descriptive case studies. The study does not explain an existing theory, so it cannot be categorized as explanatory. Rather, it tries to describe and explore emerging software development and marketing processes. Fieldwork and data collection were done prior to exact definition of the research questions and hypotheses generations.[15]

The research includes multiple exploratory/descriptive case studies, using replication logic. Replication logic is a logic by which case studies are selected to create a multiple-case design. Cases are selected so that they can either produce typical, negative or disconfirming results or exceptional/discrepant results. This form of case selection is also known as a theoretical sampling of cases as opposed to the normal sampling logic used in quantitative methods.

The outcomes of this design are improved theory, generalization ability and cross-case analysis. The latter is achieved by the use of two additional case studies which serve as control or baseline studies. Each case study is treated as an independent experiment, and the entire study is comprised of and based upon a sequence of multiple experiments.[16] When a case study strategy is agreed upon, it permits for both qualitative and quantitative sources of evidence to be collected and analyzed.[15] The collection and analysis of these two complementary forms of evidence has enabled triangulation. Various methods of data collection and fact retrieval were utilized, as described in section (c) of this chapter.

The field-study strategy, by which this research obtained insights into the processes that innovative software products go through, is an empiric study with multiple case studies. The OPCAT software was introduced into five organizations operating in mostly different vertical market sectors, so that the lessons learned from them cut across sectors. The list of sectors included banking, military, avionic, software and banking-software. One additional off-the-shelf software product (Product B) was introduced to a telecom company, and the final product (Product C) was introduced to a software organization. The number of case studies chosen (7) reflects a practical balance between the need for sufficient ground for generalization of the findings and the research time and capacity constraints. The number corresponds to the recommended range of 4 to 10 cases for theory building purposes.[15]

b. Case Study Sites

As the choice of organizations in which to perform the case studies is not a pure random sample, we tried to compensate for this by using theoretical sampling [13] designed to cover a broad spectrum of sectors, company sizes and locations, as depicted in table 2.

The customer organizations chosen for the case study sites were:

Org.1-E is a large Airborne Avionics Systems Manufacturer that employs over 3000 employees.

Table 1: Profile of firms in the exploratory/descriptive multi-firm study

Company	Sector	Size(Employees)	Location	Introduced Product
Org.4-B	Banking	>10.000	Israel	OPCAT
Org.1-E	Military/Avionics	>10.000	Israel	OPCAT
Org.2-EL	Military	<1.500	Israel	OPCAT
Org.5-Q	Software/Cellular	-100	Israel	OPCAT
Org.3-S	Software/Banking	-1.000	Singapore	OPCAT
Org.6-C	Telecom	>1.000	Israel	Non OPCAT
Org.4-B	Software	>7.000	Israel	Non OPCAT

Org.2-EL is a medium sized division (<1000 employees) of a high-technological military products manufacturer of ground, air and space-related products.

Org.3-S: Six of the seven case studies were held in the same country and the remaining study took place abroad, where the sale procedure to a large Asian banking software developer employing around 1000 people was followed.

Org.4-B is one of the largest banks in Israel, employing around 10,000 employees. The attempt was made to the bank's business software applications division.

Org.5-Q is a small software company with less than 100 employees, developing software for the cellular phone industry.

In addition to introducing OPCAT into five organizations, two additional case studies - Org.6-C, Org.7-BM - were held in which similar OTS software products were followed by means of their introductory phases into the market. These two additional case studies served as baseline case studies and assisted with both building the validity and analysis of the findings from the first five studies and with building a more robust and accurate SDLC model[4].

The case study sites were monitored periodically according to the type of evidence that was collected. For routine correspondence and product-related documentation, ongoing collection was used. For researcher observations, such as meeting attendance with adopting organizations, they were held according to the case study's natural timetable. Evidence collection sessions that is pushed by the researcher, such as questionnaires/surveys and interviews, were held at fixed time intervals across all case studies so that a matrix of observations - period versus company - was created. These mixed monitoring methods enabled the evaluation of evidence versus specific reference points in time and the description of continuous events as they were unfolding.

c. Data Collection

The software products that were introduced into each of these organizations, and the processes that they underwent thereafter until successful adoption installation and acquisition, or possibly rejection by the organization, were monitored and meticulously documented. Four different types of evidence collection were utilized for the monitoring and documentation of the above- mentioned processes: direct passive observations, documentation collection, open-ended and focused interviews and physical artifacts i.e. generated computer code or diagrams.

Table 3 summarizes the data collection efforts in the five main case studies held with the OPCAT vendor. The table also clearly indicates that the most extensive case studies as far as data collection was concerned were Org.1-E and Org.3-S. Two of the remaining 3 studies were shorter studies, mainly as they represent failed attempts of implementation by OPCAT, and thus spanned a shorter life-cycle.

[4]Due to strict non-disclosure restrictions, the information regarding the two baseline case studies, as well as the products, organizations, and customers examined has been kept confidential notwithstanding its use for hypothesis building and generalization purposes.

Table 2: General - data collection types and data collection summary statistics

Case site	Duration (Months)	Inter-views	Direct Passive Meeting Documen-tation	Question-aires	Documentation Emails/ Hard copy/ PPTs/Other	Other Physical Artifacts
Org.1-E	14	4	12+5 day brainstorming session	-	2/ 5/ 10/ 5	3
Org.2-EL	14	4	7	1	22/ 3/ 6/ -	30
Org.3-S	20	6	12+5 day session Course	0	157/ 3/ 8/ 3	8
Org.4-B	9	2	8	1	12/ -/ 4/ -	2
Org.5-Q	6	2	5	1	10/ -/ 3/ -	2

The evidence gathered from these case studies was then compared and analyzed and eventually enabled the identification and definition of common phases that the software went through in the various organizations and industries.

d. The Preliminary Model - A General View

Per the research method described above we defined our own set of a priori basic constructs for this research. The various preliminary variable groups and their interaction are modeled using Object Process Methodology (OPM), which provides a variety of complexity management tools that help diagram the model clearly and efficiently.[17] The top-most diagram, seen in Figure 4, demonstrates the main process of successful OTS software product implementation, which this research examined.

This process and its impact on the SDLC of OTS software product innovations was our dependent variable. This process is handled and impacted by the various stakeholders in today's OTS software product market, i.e., the vendor, adopting organization, third party integrators/consultants and, indirectly, other market and industry effects that constitute the intervening and contextual variables of this model, respectively.

The stakeholders interact via external non-systemic, environmental social networking process, in which they exchange leads, assign projects, etc. The impact of this process as a whole was of concern to this research, but its internal components and intricacies were not further elaborated, as the issue of social networking has already attracted extensive writing and research.[18]

The independent variable of this model is the OTS software product. For the sake of simplicity, the object representing the product includes only four basic states: specified, developed, acquired, and implemented. These are the most important states in a software product's lifecycle from the initial undeveloped product state, i.e., product in specification format only, to the successfully implemented product by an adopting organization.

Research model links in general represent possible hypotheses resulting from these relationships. Thus, the bidirectional effect links connecting the various attribute groups in Figure 4 mark the possible influence each group may have on others. The bidirectional links generalize unidirectional and bidirectional influences and suggest the variable group undergoes certain changes once the process is performed.

The preliminary model spanned 38 variables brought together from the various IS, OR and marketing domains discussed in Section 3. For the sake of brevity we do not bring here a full discussion regarding the reasons for their inclusion, and the variables comprising each variable group|footnoteThe full expla-

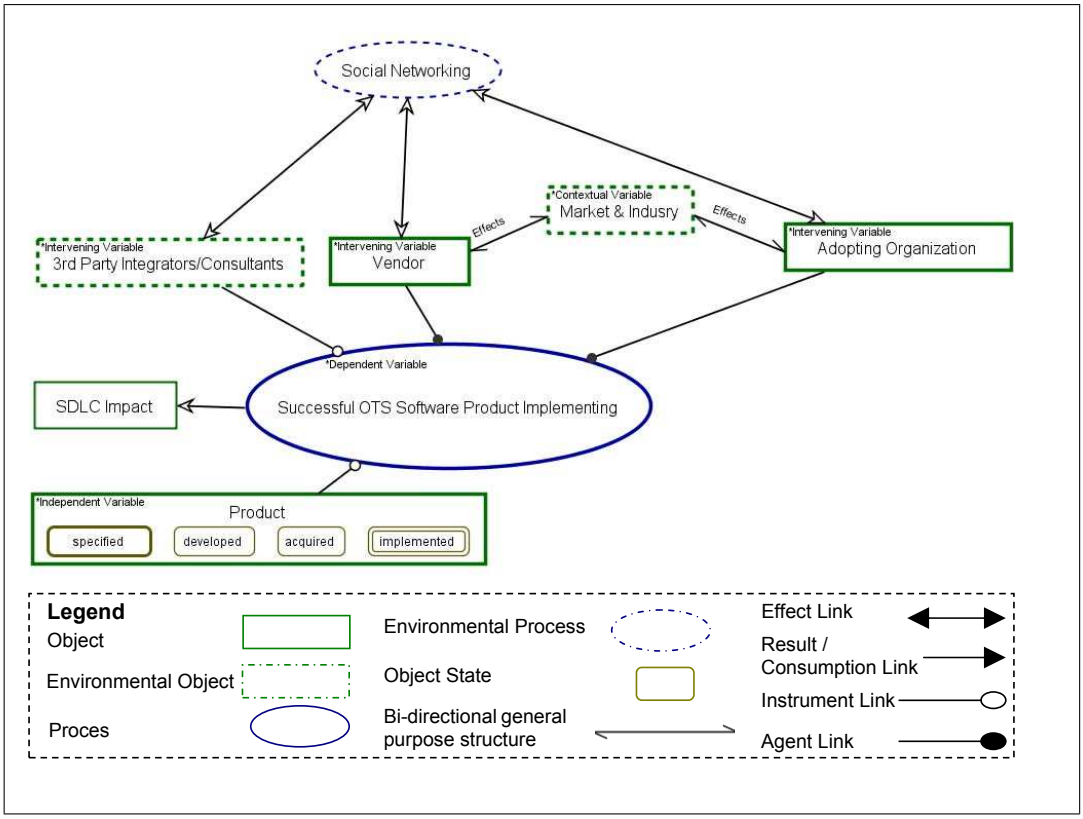

Figure 4: The proposed preliminary logical model

nation is readily available from the authors.. Furthermore, the full list of initial variables, as well the final ones, is given in table 4.

5 Intermediate Findings and Conclusions

The findings of this research include the final collaborative multi-tier SDLC research model and specifications of how the original research model has been modified throughout the research by omitting, adding and merging variable groups and variables. The aim here was to show how the model and its variables were validated through the various case studies conducted. A table is used to show the original model's variables vs. the final model's variables, and how each variable gained or lost validity based on evidence from the case studies.

Therefore, in this section, we begin by describing the changes to the original research model in (a), and continue with explaining about Lead-Driven Development (LDD) described in (b) brought as a partial downscaled example of the much larger and full Collaborative Multi-Tier System Development Lifecycle model. We end the section by giving a short explanation as to the contents of the full Collaborative Multi Tier System Development Lifecycle model in(c).

a. Changes to the Original Research Model

Following the guidelines of the case study research methodology [16], we entered the case sites using a preliminary suggested research model described in Section 4. As a case study proceeds, the research model is often updated by adding previously missed-out variables and deleting unnecessary or irrelevant ones. Table 4 lists the variables in the original and final models, their inclusion or exclusion in the preliminary and final models, the case studies upon which the exclusion or inclusion were based, and the

nature of the impact of the variable. The nature is depicted using three symbols: +, - and OT, as explained next. The plus symbol is used to denote a positive influence on the successful sale and implementation of the OTS software product. The minus sign is used to denote a negative influence on the successful sale and implementation of the OTS software product. OT, which denotes "Other" reasons, is used when the impact is of a compound or qualitative nature. The last "Based on" column provides a supporting reason from the literature for the inclusion or exclusion of a specific variable. Explanation of the unique impacts and findings of the findings mentioned above:

The point in time where the implementation process is completed has been found to be, in the discussed product type, the third sale point. This is so as an initial first sale is either an impulsive buy or an exploratory attempt and is bundled with a human implementer. The second sale is a post-sale buy to try and achieve the product's associated benefits independently of external vendor human resources attached. The third sale constitutes reconfirmation of the product's benefits to the adopting organization and is characterized by purchase of licenses and a long-term support plan.

The political factors were observed in only one case study which accidentally took part during the Second Lebanon War, which took place in Northern Israel during the summer of 2006. However, we attributed this to coincidence and did not otherwise find any political issues effecting the market or industry and therefore deleted these two variables from the final model.

A unique influence was found in two of the larger case study sites, i.e., the partially-government-owned organizations. In these sites, a very strong influence of outsourced personnel, sometimes even positioned within the adopting organization's decision making units, was noticed.

The addition of the outsourced human resources as a descriptive characteristic of the customer's users was done in tandem with the addition of the same variable in the vendor's descriptive variables. This is possible, as in many closed and highly specialized industries, many of the organizational employees today are outsourced employees, who are often sent from a common pool of HR outsourcing companies and employees.

b. Lead-Driven Development

Based on the case studies carried out as part of this research, a new approach to software development for off-the-shelf (OTS) products of entrepreneurial vendors has been identified. The new model, called Lead Driven Development (LDD) includes detailed guidelines for entrepreneurial vendors developing OTS software. These include directions for pure development procedures (at the coding level) along with organizational steps to be held in conjunction with the coding process to support successful product implementation. This model relies on and revolves around an innovative procedure of improvisation, which is new to this industry. Improvisation counters many current trends which state that increased formality yields successful implementation.

As Table 5 indicates, LDD may be highly beneficial to the vendor. Examining table 5, we see that in all the five case studies, some form of LDD was followed. The classification level of LDD correspondence of each vendor per each case study site was scored on a scale of 1 to 5, where 5 represents complete correspondence. The classification was made by a number of uninvolved parties who checked for a clear-cut correspondence of the software introduction and development process to LDD. Since we examined the development process performed by the vendors and did not follow other stakeholders, we isolated the benefit associated with the use of LDD to be the influence of LDD on OTS product sales in the corresponding case study. Benefit was therefore observed as the influence of LDD in achieving a preliminary sale with an organization (first sale). A higher level of benefit was achieving a second sale and the highest level - a third sale. As explained later, a third sale is a measure of successful implementation. Since the observed vendor is of an entrepreneurial character, associated benefits of an efficient development process, such as shorter coding times or increased flexibility were not accounted for.

Table 4. Original vs. final research model variables

Top Group	Variable Group	Variable	Original Model	Final Model	Case Studies	Nature of Impact/ Signi-ficance	Based on
Product	SDLC Maturity	Introduction Stage	✓	✓	EL,E,S,B,Q	-	
Product	SDLC Maturity	Growth Stage	✓	✓	EL,B,Q	+	
Product	SDLC Maturity	Maturity Stage	✓	✓	NA	+	
Product	SDLC Maturity	Decline Stage	✓	✓	NA	-	
Product	Trialability		✓	✓	EL,B,Q	+	
Product	Complexity		✓	✓	EL,E,S,B,Q	-	
Product	Compatibility		✓	✓	EL,E,S,B	+	
Product	Relative Advantage	Functional	✓	✓	EL,E,S,B,Q	+	
Product	Relative Advantage	Economic	✓	✓	EL,E,S,B,Q	?	
Product	Relative Advantage	Emotional	✓	×	Not in any	NA	
Product	Whole Product Factor		✓	✓	EL,E,S,B	+	
Product	Specification Flexibility		✓	✓	E,S	+	
Vendor	HR Structure	DMU/ Stakeholders	×	✓	EL,B,Q	OT	
Vendor	HR Structure	Personnel/ Outsourced HR	×	✓*	EL,B,Q	OT	[22]
Vendor	Service		✓	✓	EL,B,Q	+	
Vendor	Business Model	Lock-in	✓	✓	EL,E,S,B,Q	+	
Vendor	Business Model	Novelty	✓	✓	EL,E,S,B,Q	+	
Vendor	Business Model	Efficiency	✓	✓	EL,E,S,B,Q	+	
Vendor	Business Model	Complemen-tarities	✓	✓	EL,E,S,B,Q	+	
Vendor	Marketing Strategy-4Ps	Price	✓	✓	EL,E,S	-	
Vendor	Marketing Strategy-4Ps	Promotion	✓	✓	EL,E,S,B,Q	+	
Vendor	Marketing Strategy-4Ps	Place	✓	✓	EL,E,S,B,Q	OT	
Vendor	Marketing Strategy-4Ps	Product	✓	*✓	EL,E,S,B,Q		

Table 4. Original vs. final research model variables (cont.)

Top Group	Variable Group	Variable	Original Model	Final Model	Case Studies	Nature of Impact/ Signi-ficance	Based on
Market and Industry	PEST	Political	✓	×	EL	NA	[19]
Market and Industry	PEST	Economic	✓	✓	EL,E,S,B,Q	+	
Market and Industry	PEST	Sociological	✓	×	Not in any	NA	[19]
Market and Industry	PEST	Technological	✓	✓	EL,E,S,B,Q	+	
Market and Industry	Industry Type		✓	✓	EL,E,S,B,Q	OT	
Adopting Organization	Type	Innovators	✓	✓	EL,E,S	+	
Adopting Organization	Type	Early Adopters	✓	✓	B,Q	+	
Adopting Organization	Type	Early Majority	✓	✓	NA	+/-	
Adopting Organization	Type	Late Majority	✓	✓	NA	-	
Adopting Organization	Type	Laggards	✓	✓	NA	-	
Adopting Organization	Users	Change Resistance	×	✓	E,B	-	[20]
Adopting Organization	Users	Profession	✓	✓	EL,E,S,B,Q	+	
Adopting Organization	Users	Position	✓	✓	EL,E,S,B,Q	+	
Adopting Organization	Users	Learning Curve	×	✓	E,B	+	[21]
Adopting Organization	Users	Outsourced HR*	×	✓	B,E	OT	[22]
Adopting Organization	DMU	Key Events	✓	✓	EL,S	OT	
Adopting Organization	DMU	Time	✓	✓	EL,E,S,B,Q	+	
Adopting Organization	DMU	Power Position of Employees	✓	✓	EL,E,S,B	+	
Adopting Organization	DMU	No. of Decision Makers involved in process	✓	✓	EL,E,B,Q	-	
3nd Party Integrator/ Consultant	Outsourced HR*		×	✓	B,E	OT	[22]

Table 5: Level of Lead-Driven Development implemented by vendors vs. Benefit in sales.

Site	LDD Level	Sale	2nd Sale	3nd Sale
Org.1-E	4	+	+	+
Org.2-S	3	+	+	-
Org.3-EL	5	+	not yet	not yet
Org.4-Q	1	-	-	-
Org.5-B	1	-	-	-

We entered the more comprehensive table figures of table 5 into a statistical software tool and found that the correlation between sales, second sales and third sales (i.e. vendor benefit) and the level of LDD implementation is clearly significant, positive and high.

After establishing that the use of LDD is beneficial for the vendor we carefully documented this process and generalized it over all the case studies. The description of the full LDD process now follows.

The hereunder elaborated emerging process for software development includes 12 main steps of which at least 4 include some form of improvisation. Moreover, the most unique phase of this suggested model, the lead gathering task is improvisation intensive. In addition, the model may serve as a strict continuous model similar to the Waterfall model or may be used as a Spiral model involving repetitive tasks.

At the core of this model, are 12 steps as follows:

Step 1: Initiation - This stage is a formal stage in the regular standard SDLC model. However, with entrepreneurial firms this step tends to be an informal one with no accurate start point in time. This stage includes structuring the will and intent to begin with the project and giving the go-ahead instruction as well as providing the limited resources necessary to start exploring the venture.

Step 2: High Level Concept Development - This second step includes forming the high level concept of the product, the problem which it comes to solve, its associated benefit etc. Depending on the scope of the project/system this development phase may be fulfilled using limited resources, spare time, and sometimes even academic resources. A substantial level of improvisation is used at this early stage as well - as part of the founding process.[23] Improvisation is carried out in the development of the suggested product in a quick and result-oriented fashion while using minor or no documentation and testing at all.

Step 3: Prototype - The first important milestone of the entrepreneurial vendor is the ability to deliver a functional prototype. The prototype should convey clearly the problem it is solving, its abilities and associated benefits in an easy and understandable manner with a friendly user interface. The number of moderate bugs, missing features as well as load balancing issues is not of much importance at all at this stage as the product shall be used mostly for demo and pilot purposes in the near future. This first prototype release is called by us a "Bugged Release". The prototype should further include a number of working examples from various domains.

Step 4: Minor Testing in Non Profit Environments, Academic Demo and Use - After completing the prototype, which should by this time be a powerful demonstration tool, the vendor should strive to demonstrate the tool in non-profit environments. The aim of these demonstrations is finding a limited installation bed for the product. These installations provide the developers with important feedback on bugs, missing features and general use of the tool - a preliminary focus feedback group for the tool. The academic scenery is extremely beneficial for these purposes as it also hosts great uncovered commercial potential through conventions, conferences and to a vast number of current or to be professionals. See also penetration attempts in academia by established firms like Philips, IBM and SAP.

Step 5: Market Introduction, Benefit Oriented Demonstrations and Mini Pilots - An additional task which is improvisation intensive is demonstrating the tool to potential customers. The suggested form of product demonstration which we call "benefit oriented demonstration" is a special type of marketing method unused so far in the world of software. The equivalent in the non-software world is that of a vacuum cleaner demonstration in the customer's home to show him immediate benefits of the product.[24] Thus, in this situation we suggest vendors demonstrate their new tools by implementing a form of improvisation at the customer's site. The vendor should use the tool to perform an on the spot real work task brought in by the customer for which neither the vendor nor the customer were prepared. If this session exceeds one meeting it may be considered as a mini pilot.

It is also in this step that the main improvisational task of the entire proposed model is undertaken. From meeting to meeting the vendor's marketing representatives must try to anticipate - using preliminary talks, phone conversations, social networking ties or emails - the needs of the potential customer

as well as his existing environment. This highly informal improvisational task is used to build software requirements for the development coding team.

The requirements gathered are for features which will be required in the marketing sessions with the potential customers. These requirements are then addressed and coded immediately by the development team. The new features developed are neither documented nor tested thoroughly as they will be mainly used for demonstration purposes and may be ultimately dropped. However, a mentioning and documentation of the added features is required at least in a "What's New" file accompanying the product.

This step is of a strict repetitive improvisational nature and involves the gathering of lead requirements between marketing meetings and converting them into semi-operational software features.

Step 6: Offer OTS + HR = Project. After one of the leads materializes the vendor is asked to prepare a formal proposal for sale. Many of our case studies have indicated that OTS products for the professional organizational realm are rarely sold if they come from unknown vendors due to risk factors mainly of product abandoning. Hence, offering the OTS with Human Resources (an implementer) which will assist the adopting organization, prepare the initial material and then tutor its users is usually beneficial to reduce the uncertainty in these situations. After the customer agrees upon the terms of the project (and not only the product) contract engaging commences and the customer is now considered as the baseline customer. Hence, we call this sale the "first sale" en route to successful implementation.

Step 7: Bug fixing due to baseline customer requirements and marketing requests - After a certain amount of work is done using the tool at the customer's site, either by the customer's users or by the HR which was coupled to the OTS, important feedback regarding the products begins to accumulate. This information enables bug fixing and tool robustness improving. In addition, important missing features required by the baseline customer and marketing department are added to the software and provide for the First Commercial Release of the software. This release is still highly saturated by bugs but is already a commercial useable - "non-frustrating" - version of the product.

Step 8: Constrain Features, Further Commercial Releases and Support Plan - The unbridled adding of new features, in the format suggested in steps 5 and 7 above, creates overwhelming monstrous software. At this point the vendor should start to funnel out some features which cater to a smaller audience and which have not been found to be part of the vendor's targeted audience needs. Furthermore, the vendor should try to find a common thread or theme connecting and guiding all other features. This decision enables further product releases each containing additional noteworthy features, bugs correction and feature enhancements. With the continuous use of the product in at least one baseline customer and before the move to the next implementation step, a support plan (or plans) for the product should be created.

Step 9: Develop complexity management tools, Train integrators personnel, Interface with customer's software, Find additional benefit oriented projects, Embed within organizational deliverables - Within the baseline customer's everyday work, issues of model complexity very quickly arise. These issues, which are different from testing the product or load-balancing it, should be addressed and solved early on. In addition, this is also the time to deepen the roots within the customer by both trying to embed the product deliverables within the customer's overall deliverables, interfacing (physically) with the company's organizational ISs and by finding additional projects within the customer's company to be involved with. Deepening the vendor's role within the customer's site is a highly improvisational task in nature and hence requires adequate skills.

Step 10: Second Sale - Licenses: The first sale is by no means any indication of a successful implementation or diffusion of the innovation within the adopting organization.[25] Moreover, the coupling of the HR with the OTS does not really enable a real diffusion of the independent product. Hence, a second sale to the same organization marks an important future commitment of the organization to drop the tutoring relationship and proceed to license purchases for independent use. License purchases signify that the organization now associates positive benefit to the use of the product.

Step 11: Maintenance, Begin User Training, Tutoring, Software Support - After a second sale is

made, the relationship between the vendor and customer moves into the maintenance phase. As the previous phase was coupled by HR this is where the customer will be taking his first steps with the software alone. These first steps include: formal user training sessions, one on one user tutoring sessions and general software support.

Step 12: Third Sale, Support - The earliest point one might consider as the point of successful diffusion or implementation of the innovation is, as seen in this study, the point of third sale. After the second sale, the customer independently used the software by himself, learnt the product's advantages and disadvantages and may now associate self benefit to product more accurately. Therefore, a third sale is the first true mark that the customer is truly realizing the benefits of the product and is preparing to use it in the long term. Support of the product continues now on an annual basis with milestones for new upgrades and releases.

Given that a vendor adopts this new 12 stage development process including and especially the improvisation-intensive stages, the question which should arise is how does the organization build and enhance the skills required for improvisation and what are these skills?

In the context of our study we identified three main factors which influence improvisational skills:

Teamwork skills - The ability of the entrepreneurial team to communicate with one another and relay timely information, get things done easily and quickly with no inhibitors and outside impeding factors.

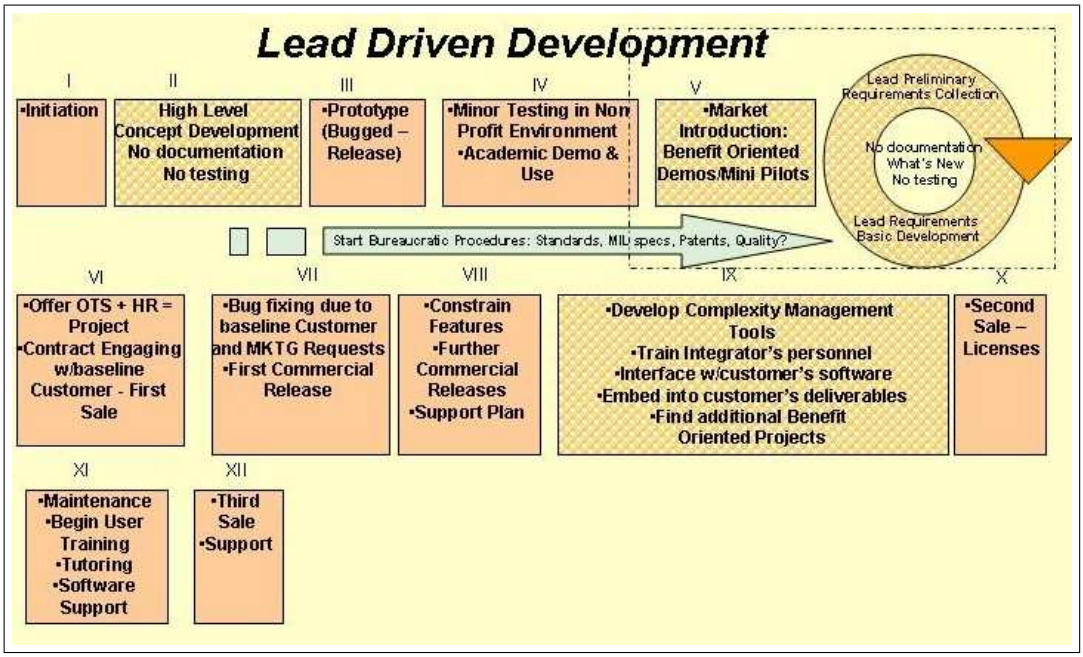

Figure 5: Lead Driven development: the 12 step timeline

Experience - The entrepreneurial team members' experience in similar circumstances and their memory to recall their right and wrong doings there.

Experimental culture - The culture of the team, which encourages trying out many a time risky and/or innovative solutions.

These three main factors, measured and calibrated according to characteristics described in [26], coincide with improvisational skills factors in the literature. For example, Vera and Crossan [27] create a theoretical framework based on improvisation and innovative performance in teams. Identifying variables from improvisational theatre, they tested the impact of the 16 different related variables on improvisational skills in an environment of a local municipality. They found 4 of the 16 variables to be of higher influence than others. The four factors they isolated were: expertise, teamwork skills, experimental culture and real-time information and communication.

In the context of our study we can therefore translate and apply their insights as follows: Expertise - Gaining a higher level of software expertise in the intricacies of the development environments enables software developers to find out of the sleeve solutions and bypasses for many software unpredicted difficulties encountered.

Teamwork skills - In general software teams, with a rather higher sense of collaboration usually tend to innovate more. To further clarify this point, we can propose the software teams the following teamwork skills which we encountered: development collaboration, information sharing via email, shared drives, knowledge management portals, inner group dynamics and communication etc.

Experimental culture - The experimental culture includes the ability to import new ideas and procedures from the World Wide Web, forums, groups and software development associates and try them out. Furthermore, experimentation in the software industry, which is not usually backed up by management should be backed up by top management and should also include experimentation on code developed using a number of alternate mechanisms.

Real-time information and communication - The need for real-time information and communication in the software industry is ever more compelling than any other industry. This is so because the software industry is built upon and relies heavily on the backbone of internet. Therefore, when improvising it is crucial to gain real-time updated information over the LAN or internet and have a variety of channels for communicating with the customer and other team members. Each of these channels specializes in a different type of content that may be passed: audio, video, documents, emails etc.

c. The Collaborative Multi-Tier System Development Life Cycle

The Lead-Driven Development (LDD) paradigm described above represents a list of tasks from the vendor's point of view. This list helped build the vendor's tier in the complete Collaborative Multi-Tier System Development Life-Cycle task matrix that caters to all stakeholders. The Collaborative Multi-Tier model takes into consideration, through the nature of the tasks suggested, pure IS development tasks (e.g., prototyping, bug fixing), market influences (e.g., market introduction techniques for entrepreneurial OTS software vendors, such as offering the first sale of such a product as a combined project with human resources), and organizational recommendations aimed mainly to avoid customer internal organizational obstacles. The Collaborative Multi-Tier System Development Life-Cycle (CMSDLC) model, which combines common tasks from the various case studies, is consistent as it avoids contradicting tasks. One of its unique features is that it makes a distinction between benefit-oriented tasks and standard waterfall-type formal tasks and milestones.

When a task or a set of tasks is performed in an iterative manner it is marked as an on-going task or specifically mentioned in the explanation section as one that needs to be performed iteratively. One such example is the task dealing with lead requirements gathering from potential customers for the purpose of prototype and feature building.

The CMSDLC model is also suitable for mature organizations that seek to develop a new OTS software product. It is even more suitable when the mature organization separates this entire operation from its existing core operations through various methods, such as founding a new subsidiary. This is akin to an entrepreneurial firm from the development and market perspectives. However, the financial backing of the parent organization and its reputation may shorten the duration of many of the tasks, and may make them more easily achievable. In cases where a mature organization seeks improvement to existing product development methods, the suggested model may not be as applicable, since the level of uncertainty such an organization encounters regarding market and organizational effects (especially regarding the customers), is lower.

The chart of the full model depicts in a single, poster style view the multi-tier model. The timeline of the system development lifecycle and the tasks for each stakeholder are stacked on separate horizontal lines. Since all the stakeholders share one timeline, the interaction between the stakeholders and the interdependency of tasks is clearly visible. Only through extensive collaboration can the effort proceed

as a whole. For each task, a list of case studies which the task was based upon is provided. A list of one-letter abbreviations corresponding to the case study sites which demonstrated the use of the specific task appear below each task.

The logic behind gathering all the tasks into the model includes checking tasks for contradictions in the various case studies and merging similar tasks to unifying generic tasks.

6 Conclusion and Future Research

At this point we would like to introduce the hypotheses which were derived from the findings of our case studies. The hypotheses suggest a plethora of possible future research to be held on the process of validating them.

The first hypothesis which was emerging was based on research question 2. We found that the two parts of an OTS software product, the product and its underlying methodology, played an important role in all the case studies and therefore justified the hypothesis. We therefore phrased the first hypothesis as follows.

H1: *The more a customer is inclined to adopt a methodology, the more he or she is inclined to adopt the OTS software tool associated with that methodology.*

In other words, we will be looking for a possible correlation on a "bundling" relationship between the methodology and its supporting tool. We can explain this phenomenon by looking both at the marketing literature regarding characteristics of products and at the IS literature which explains and recommends various modes for IS product sale.

The marketing literature [28] and [12] and [29] defines clearly the "whole product concept" which was introduced by Theodore Levitt.[30] Levitt defined the whole product as follows: A product is, to the potential buyer, a complex cluster of value satisfactions.

The whole product factor denotes the completeness of the product being marketed at a certain point in time with regard to a complete solution. The components of the whole product include the core product, the tangible product, augmented product and total solution. The relationship between the OTS product used in our study and its methodology typifies the product as being close to an augmented product.

The second hypothesis involved a distinction between organizations using a formal software development procedure and others who use improvisation-intensive development techniques. As we started noticing in the case studies, this distinction is related to the level of uncertainty the vendor organization runs into. Hence we defined the following hypothesis:

H2: *The higher the level of uncertainty is, the higher the vendor's use of improvisation intensive development methods.*

Improvisation has been shown to be used by entrepreneurial vendors when faced by time pressure, complexity, and uncertainty.[31] Future research may extend this assertion to include and emphasize its relevance in the development process too.

The third hypothesis concerns the direct benefits associated with utilizing improvisation within the software development process, i.e., the time-to-market of the product and increased sales. The hypothesis was defined as follows.

H3: *When uncertainty is high, the more a vendor uses improvisation the more his market response time shortens and his ability to make a first sale improves.*

The organizational change resistance to new technologies in general and software in particular was found in our case studies to be solved by applying a marketing technique which couples a human resource implementer to the product.[32] The human resource implementer escorts the implementation and even performs most of the initial work for the customer using the tool. This triggered the following hypothesis:

H4: *The higher the level of HR participation in the OTS software sale attempt to the change resistant customer is, the higher are the chances for successful implementation.*

Similarly to the entrepreneurial vendor's efforts to overcome change resistance within the adopting organization users, the vendor has to build its legitimacy with the adopting organization's DMU.[33] We started noticing that this legitimacy buildup was being done via affiliation with accredited scientists/academics and/or through established third party integrators.

Stinchcombe [33] also specified three reasons for the impediments companies have from entering into a business relationship or buying a product from a new organization. He calls this effect the "Liability of Newness". The reasons cited for this liability are Lack of Experience, Lack of Size and Lack of Legitimacy. The latter was addressed by OPCAT, who built its legitimacy in all the case studies through the use of a reputable scientist to improve its lack of legitimacy and external reputation. Furthermore, some legitimacy build-up was achieved through the use of large third party integrator with proven reputation and experience in the industry.

This gave rise to the following hypothesis:

H5: *The more a new vendor firm affiliates with a distinguished scientist, or an established third party, the more the chances for successful implementation are higher.*

7 Summary and Recommendations

We have proposed and evaluated a software system development life cycle model which aims to improve successful system implementation and adoption by use of communication and collaboration amongst stakeholders. The new model for software development that emerged - Lead-Driven Development - was discovered, validated against the case studies, and refined via observations in five industry case studies and two additional control studies regarding successful implementations of OTS products of entrepreneurial developers.

The proposed Lead-Driven Development model accounts for market and organizational factors and the way they are woven into the traditional phases of software development. It offers the basis for the unified, comprehensive multi-tier SDLC framework and methodology that contributes to improved stakeholders' communication and collaboration through the use of a common reference model for all stakeholders. Each tier addresses a different force or stakeholder involved in the software market: vendor, customer, consultants and integrators.

The model is potentially beneficial for improving communication and collaboration among life cycle stakeholders in that it embeds action items from the IS, marketing and organizational realms. Many of these action items are performed using improvisational skills.

To excel in Lead-Driven Development in general, and in software development improvisation in particular, entrepreneurial vendors should enhance their improvisational skills. In line with previous studies [27], we found three main factors that influence improvisational communication skills: experience, teamwork skills, and experimental culture. Focusing on these facets, organizational training should provide a clear positive effect on improvisational skills and hence on innovation abilities.

Bibliography

[1] S. Sawyer, A market-based perspective on information systems development, *Communications of the ACM* (44:11), pp.97-102, 2001.

[2] E. Carmel, Cycle Time in Packaged Software Firms, *Journal of Product Innovation*, (12), pp.110-123, 1995.

[3] D. Dori, *Object Process Methodology*, Springer-Verlag, 2002.

[4] W. Royce, Managing the development of large software systems, In *Proceedings of IEEE WESCON*, pp.1-9, 1970.

[5] *TechTarget*, viewed on http://searchvb.techtarget.com/sDefinition/0,,sid8_gci755068,00.html, February, 2004.

[6] D. Howe (Ed.), *The Free On-line Dictionary of Computing*, http://foldoc.doc.ic.ac.uk/ foldoc/-foldoc.cgi? query=lifecycle, 1996.

[7] *GAO, GAO/IMTEC-8.1.4 Assessing Acquisition Risks*, http://www.gao.gov/ special.pubs/im814.pdf, 1992.

[8] B. Krogstie, and B. Bygstad, Cross-Community Collaboration and Learning in Customer-Driven Software Engineering Student Projects, *CSEET Proceedings of the 20th Conference on Software Engineering Education & Training*, 2007.

[9] E. Carmel, and S. Becker, A process model for packaged software development, *IEEE Transaction Engineering Management*(41: 5), pp.50-56, 1990.

[10] M. Cusumano, A. MacCormack, C. F. Kemerer, and B. Crandall, Software Development Worldwide: The State of the Practice, *IEEE Software*, pp.28-34, 2003.

[11] M. Keil, and E. Carmel, Customer-developer links in software development, *Communications of the ACM* (38:5), pp.33-44, May 1995.

[12] G. Moore, *Crossing the Chasm*, Harper Collins, 1999.

[13] R. K. Yin, The Abridged Version of Case Study Research: Design and Method, In L.Bickman and D. J. Rog *Handbook of Applied Social Research Methods*, Thousand Oaks, CA: Sage, pp.229-259, 1998.

[14] W. Tellis, Introduction to Case Study, *The Qualitative Report* (3:2) July 1997. (http://www.nova.edu/ssss/QR/QR3-2/tellis1.html)

[15] K. M. Eisenhardt, Building Theories from case Study research, *Academy of Management Review* (14:4), pp.532-550, 1989.

[16] R. Yin, *Case Study Research: Design and Methods*, Thousand Oaks, London, New Delhi: Sage, 1984.

[17] D. Dori, I. Reinhartz-Berger, and A. Sturm, OPCAT - A Bimodal CASE Tool for Object-Process Based System Development, *Proc. IEEE/ACM 5th International Conference on Enterprise Information Systems* (ICEIS 2003), École Supérieure d Électronique de l Ouest, Angers, France, pp.286-291, April 23-26, 2003.

[18] C. Typaldos, *Social Networking*, viewed on http://www.typaldos.com/, 2003.

[19] *PEST*, viewed on http://www.marketingteacher.com/Lessons/lesson_PEST.htm, November 2004.

[20] K. R. Stam, J. M. Stanton, I. R. and Guzman, Employee Resistance to Digital Information and Information Technology Change in a Social Service Agency: A Membership Category Approach, *Journal of Digital Information*, vol. 5 Issue 4, 2004.

[21] C. F. Kemerer, How the Learning Curve Affects CASE Tool Adoption, *IEEE Software*, v. 9, pp.23-8, 1992.

[22] M. Lacity, and L. Willcocks, Information Technology Sourcing Reflections, *Wirtschaftsinformatik*, Special Issue on Outsourcing, vol. 45(2), pp.115-125, 2003.

[23] T. Baker, A. S. Miner, and D. T. Eesley, Improvising firms: bricolage, account giving and improvisational competencies in the founding process, *Research Policy*, vol. 32, Issue 2, pp.255-276, 2003.

[24] *Vacuum marketing strategy*. Viewed on http://www.marketingsurvivalkit.com/advertising-sales-strategy.htm, October 2006.

[25] R. G. Fichman, and C. F. Kemerer, The illusory diffusion of innovation: An examination of assimilation gaps.Information Systems Research, *Information Systems Research* (10:3), pp. 255-275, 1999.

[26] S. Cohen, *A Multi-Tier System Development Life Cycle Model for Off-the-Shelf Software with Market and Organizational Effects. Ph.D. dissertation*, Faculty of Industrial Engineering and Management, Technion, Haifa, Israel, 2007.

[27] D. Vera, and M. Crossan, Improvisation and Innovative Performance in Teams, *Organization Science*, vol. 16, No. 3, pp. 203-224, May-June 2005.

[28] A. N. Alderman, Implementing the Whole Product Concept in Strategic Sector Marketing, *Proceedings of the Fifteenth Annual IEEE Applied Power Electronics Conference and Exposition*, New Orleans, LA, USA, pp.27-30, 2000.

[29] *JWR*, viewed on: http://jwr.strategictechnology.com/ pages/ChoosingSoftware.pdf, November 2005.

[30] T. Levitt, Marketing Intangible Products and Product Intangibles, *Harvard Business Review*, vol. 59 No. 3, pp.94-102, 1981.

[31] S. L. Brown, and K. M. Eisenhardt, The Art of Continuous Change: Linking Complexity Theory and Time-Paced Evolution in Relentlessly Shifting Organizations, *Administrative Science Quarterly*, vol. 42, No. 1, pp. 1-34, 1997.

[32] S&M, *How software firms spend their S&M dollars?*, http://www.softletter.com/PDFs/FHp10-p13.pdf, 2006.

[33] A. L. Stinchcombe, *Social structure and organizations In James G. March (ed.), Handbook of organizations*, Chicago: Rand McNally, 1965.

[34] S. Singh, and P. Kotze, An overview of systems design and development methodologies with regard to the involvement of users and other stakeholders, *Proceedings of SAICSIT*, pp.37-47, 2003.

[35] M. Cusumano, *The Business of Software: What Every Manager, Programmer, and Entrepreneur Must Know to Thrive and Survive in Good Times and Bad*, Free Press, 2004.

[36] K. B. Clark, and S. C. Wheelwright, *Managing New Product and Process Development: Text and Cases*, New York: Free Press, 1993.

[37] K. B. Clark, and S. C. Wheelwright, *The Product Development Challenge: Competing through Speed, Quality, and Creativity*, Boston: Harvard Business School Press, 1995.

[38] S. C. Wheelwright, and K. B. Clark, Accelerating the Design-Build-Test Cycle for Effective Product Development, *International Marketing Review* (11:1), pp.32-46, 1994.

[39] B. Boehm, and P. Bose, A Collaborative Spiral Software Process Model Based on Theory W, *Proceedings, 3rd International Conference on the Software Process, Applying the Software Process*, IEEE, 1994.

[40] C. H. Fine, *Clockspeed: Winning Industry Control in the Age of Temporary Advantage*, Perseus Publishing, 1999.

[41] D. Avison, and G. Fitzgerald, Where Now for Development Methodologies?, *Communications of the ACM*, vol. 46, no. 1, pp.78-82, 2003.

[42] C. Ebert, Requirements Engineering: Understanding the Product Life Cycle: Four Key Requirements Engineering Techniques, *IEEE Software*, pp.19-25, May/June 2006.

[43] E. Mustonen-Ollila and K. Lyytinen, Why organizations adopt information system process innovations: a longitudinal study using Diffusion of Innovation theory, *Information Systems Journal* (13:3), pp.275-297, July 2003.

[44] H. C. Jr. Lucas, and V. Spitler, Technology Acceptance and Performance: A Field Study of Broker Workstations, *Decision Sciences Journal*, (30:2) 1999.

[45] F. D. Davis, Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology, *MIS Quarterly*, pp.319-339, 1989.

[46] G. C. Moore, and I. Benbasat, Development of an Instrument to Measure the Perceptions of Adopting an Information Technology Innovation, *Information Systems Research* (2:3), pp.192-220, September 1991.

[47] J. C. Brancheau, and J. C. Wetherbe, The Adoption of Spreadsheet Software: Testing Innovation Diffusion Theory in the Context of End-User Computing, *Information Systems Research* (1:2), pp.115-142, 1990.

[48] R. B. Cooper, and R. W. Zmud, Information Technology Implementation Research: A technological Diffusion Approach, *Management Science* (36:2), pp.123-139, 1990.

[49] J. Verville, and A. Halingten, An investigation of the decision making process for selecting an ERP software: the case of ESC, *Management Decision* (40:3), pp.206-216, 2002.

[50] P. Nelson, W. Richmon, and A. Seidmann, Two Dimensions of Software Acquisition, *Communications of the ACM*, (39:7), pp.29-35, 1996.

[51] J. Iivari, and I. Ervasti, The impact of alternative IS acquisition options upon the IS implementation and success, in *Proceedings of the 1992 ACM SIGCPR conference on Computer personnel research*, Cincinnati, Ohio, United States, pp.338 - 349.

[52] M. C. Paulk, Structured Approaches to managing Change, *Crosstalk: The Journal of Defense Software Engineering* (12:11) November, pp.4-7, 1999.

[53] Nordman, *Commercialization Success in Early Stage Technology Companies*, viewed on www.rocketbuilders.com/ commercialization/ RB_Commercialization_Presentation_Jun2004.pdf, 2004.

[54] Y. Lee, and G. Colarelli O'Connor, New Product Launch Strategy for Network Effects Products, *Journal of the Academy of Marketing Science*, (31:3), pp.241-255, 2003.

[55] E. Montaguti, S. Kuester, and T. S. Robertson, Entry Strategy for radical product innovations: A conceptual model and propositional inventory, *International Journal of Research in Marketing*, (19), pp.21-42, 2002.

[56] S. Shane, and S. Venkataraman, The promise of entrepreneurship as a field of research, *Academy of Management Review*, vol. 25, No. 1, pp.217-226, 2000.

[57] S. Shane, *Entrepreneurship: A Process Perspective*, South-Western College Pub; 1st edition, January 2004.

[58] F. Murray, and M. Tripsas, The Exploratory Process of Entrepreneurial Firms: The role of purposeful experimentation, *Advances in strategic management*, vol. 21, pp.45-75, 2004.

[59] *NIH System Development Life Cycle (SDLC) IT Security Activities Matrix*, viewed on http://irm.cit.nih.gov/ security/nih-sdlc.html, December 2006.

[60] DOJ, *The Department of Justice Systems Development Life Cycle Guidance Document*, viewed on http://www.usdoj.gov/ jmd/irm/lifecycle/table.htm, January 2003.

[61] *Agile software development: a definition from Whatis.com* - Viewed on:http://whatis.techtarget.com/ definition/0,,sid9_gci936457,00.html, November 2006.

Shalom Cohen (b. December 16, 1973) completed his MSc in Operations Research at Tel Aviv University, Israel and his PhD in the Faculty of Industrial Engineering and Management at the Technion, Israel. In addition to Lecturing at the Technion on Information System related topics Shalom holds the position of Chief System Architect in a high tech company in the homeland security sector.

Dov Dori (b. September 2, 1953) is information and systems engineering Professor at the Faculty of Industrial Engineering and Management, Technion, Israel, and Research Affiliate with the Engineering Systems Division at Massachusetts Institute of Technology. He received his BSc in Industrial Engineering and Management from Technion in 1975, MSc. in Operations Research from Tel Aviv University in 1982 and PhD in Computer Science from Weizmann Institute of Science in 1988.

Uzi de Haan (b. September 5, 1943) completed his MSc in Aeronautical Engineering at the University of Delft, Holland and his PhD in the Faculty of Industrial Engineering and Management at the Technion, Israel. He joined the Technion as a professor in the area of Strategic Management and Entrepreneurship at the Faculty of Industrial Engineering after many years in the high-tech industry.