The Role of Improvisation in Off-the-Shelf Software Development of Entrepreneurial Vendors

Uzi de Haan and Shalom Cohen
Technion, Israel Institute of Technology,
Haifa, Israel
{uzid, shalom1@technion.ac.il}

Abstract

Improvisation has been discussed to some extent in the Innovation and Entrepreneurship literature with regard to processes and tasks performed by entrepreneurs when faced with uncertainty, such as in organizational learning in uncertain technology environments [9], in the decision making process [7], and in the founding process [2].

The role of improvisation has also been discussed with regard to software organizations and software development and the various tasks performed by the development heads and teams [3].

Hence, to make this connection between the software development process, improvisation and entrepreneurs, this research reports the observations made in a multiple case study scene of an entrepreneurial vendor of Off-the-Shelf software. Through these observations, a new form of software development emerged. This form, Lead-Driven Development, is based upon improvisation of the software development processes by entrepreneurial vendors who improvise new product features and bug-fixes based on leads they pick up from one marketing meeting or demonstration to the next one. This type of development, although not very beneficial in the long run for the product's commercial development tailors demo features specifically to potential customers' needs. Thus, the chances to satisfy potential customers' needs through the product's associated benefits and features, improve. We discuss in length how the improvisation is employed and recommend a software development task path based upon it. We bring figures to demonstrate how Lead Driven Development has been found to be beneficial to vendors.

Following the suggested model for software development for entrepreneurial vendors, we introduce a theoretical framework from the Innovation literature [11] and by the use of knowledge gathered from the case studies conducted, we explain how vendors may achieve an improved level of software development improvisational skills.

1. Introduction

1.1 Improvisation in entrepreneurial firms

Scholars have identified improvisation as an important organizational process [13, 8, 14]. Miner et al. (2001) define improvisation as occurring when the design and execution of novel converge. Improvisation as an organizational process is in particular relevant in the entrepreneurial founding process, which is characterized by uncertainties in markets, product development and organizational competencies [7]. It is even more so in knowledge intensive new ventures where in the early era of technological ferment uncertainty is high [9].

The dominant image of the founding process in entrepreneurship literature paints a linear process of design and then execution [16]. In the improvisation framework, founders and R&D managers may design firms and new products as they create them. Baker et al. (2003) found in their study of 58 start-ups, of which 15 were computer training firms and 21 were business to business software firms that improvisational processes permeate entrepreneurial activity and that improvisation capability is a competence which a firm may acquire. In addition, they found in their study that some software firms became proficient in problem-driven and opportunity-driven improvisational competencies. Vera and Crossan (2005) showed that teams can be trained in improvisational competences and that teams with expertise, teamwork skills, experimental culture and real-time information and communication were better in improvisation and performed better in innovative task settings

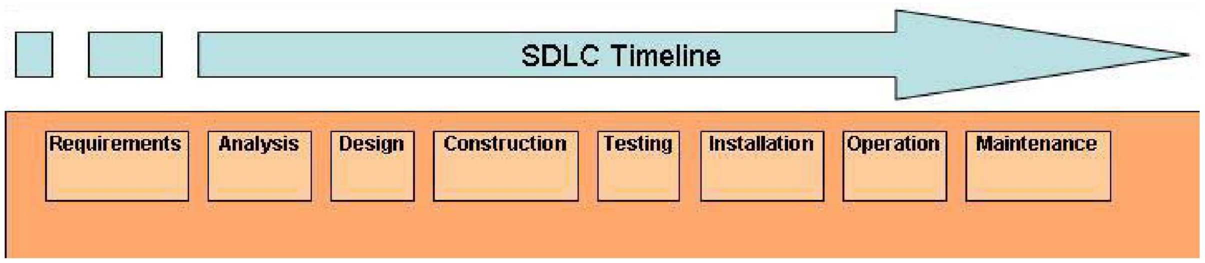

Figure 1: The basic SDLC Timeline Waterfall Model

1.2 Improvisation in the software development processes

Basic methodologies for software development usually revolve around the very basic System Development Life Cycle Waterfall Model [12] as depicted in Figure 1.

The basic steps of this approach usually involve strict software development steps beginning with gathering the customer's system requirements for the purpose of building the software to maintenance and end of life decision points. However, this basic model not only neglects two very important issues (explained hereafter) but is also missing many important phases for certain situations, in which some of its steps are even erroneous. This occurs mainly due to the basic model's aims to serve as a flexible baseline model for methodologies upon which to build more elaborate development phases. The first unaccountable issue is that the waterfall's basic steps are supposed to cater to all or most types of software systems built by all types of developing organizations. The second is that the model does not account for additional exogenous factors which impact the various phases and alter them. Exogenous factors of this sort include organizational and market effects.

Based on seven case studies carried out and described below, a new approach to software development for Off-the-Shelf (OTS) products of entrepreneurial vendors has been identified.

The new model, named Lead Driven Development includes elaborate guidelines for entrepreneurial vendors developing OTS. These steps include directions for pure development (coding) procedures as well as additional organizational steps to be held in conjunction with the coding process to the end of successful product implementation. This model relies heavily and revolves around an innovative procedure of improvisation, which is new to this industry and is contrary to current trends which state that increased formality yields successful implementation.

The use of formalism in software development, via enhanced documentation, quality and maturity certification etc., may increase benefits in a standard long-term project and may decrease chances of errors in quality or scope from occurring. However, when the time of design and execution converge and uncertainty increases there is no time for formalism.

In software application development, agile software development (ASD) is a methodology for the creative process that anticipates the need for flexibility and applies a level of pragmatism into the delivery of the finished product. Agile software development focuses on keeping code simple, testing often, and delivering functional bits of the application as soon as they're ready. The goal of ASD is to build upon small client-approved parts as the project progresses, as opposed to delivering one large application at the end of the project [1].

However, flexibility and fast response times are achieved by a flexible application of a set of preprogrammed responses and do not cover uncertainties in technologies and markets which, combined with the need for fast response times, require improvisation in addition to flexibility.

Hence, to make this connection between the software development process, improvisation and entrepreneurs, this research reports the observations made in a multiple case study scene of an entrepreneurial vendor of Off-the-Shelf software. Through these observations, a new form of software development emerged. This form, Lead-Driven Development, is based upon improvisation of the software development processes by entrepreneurial vendors and is discussed in Section 3.

Following the suggested model for software development for entrepreneurial vendors, we introduce a theoretical framework from the Innovation literature [11] and using knowledge gathered from the case studies conducted, explain how vendors may achieve an improved level of software development improvisational skills.

Research method is described next.

2. Method: About the case studies conducted

The System Development Life Cycle of an entrepreneurial vendor of an OTS product was observed in seven different case studies. Five of the case studies belong to one vendor and the remaining two represent the second vendors. All seven studies were strictly held in the commercial market spanning different sectors and different company sizes. The selection of the case study sites was made per Yin's recommendation for theoretical sampling of contrasting sites, sites of special interest etc. [10].

Table 1 - Case studies held, industry and vendor

Site	Industry	Vendor
Org.1 - E	Avionics	1
Org.2 - S	Software/Banking	1
Org.3 - EL	Military	1
Org.4 - C	Telecom	2
Org.5 - BM	Software	2
Org.6 - Q	Software	1
Org.7 - B	Banking	1

The first vendor was a young entrepreneurial company, developing an OTS software modeling tool for Systems Engineers. Five different sale attempts were followed with this vendor in organizations ranging from Org..6-Q, a small software company with less than 100 employees developing software for the cellular industry, to Org.1-E, a large Avionics Systems Manufacturer which employs over 3000. The second vendor, an OTS software developer of CAD software, was followed in two large organizations. The first, Org.4-C, a telecom system developer for the large Telecom providers, and the second, Org.5-BM, a large software development company. The authors personally joined on all sale attempts and gained access to all necessary data in the above-mentioned organizations. 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, Org.2-S, was followed.

We followed Yin's methodology for conducting case study research for exploratory case studies [10]. The organizations were observed over a period of 14 months during which four types of data collection methods were used: direct passive observations, document obtaining, physical artifacts (e.g. software code) and interviews.

The research model used for conducting the case study research is seen in Figure 2.

At the basis of this model we are checking to see how the vendor, adopting organization, 3rd party integrators/consultants as well as additional contextual variables influence the successful implementation of the OTS software product. The Software Product itself serves as our independent variable. The Successful

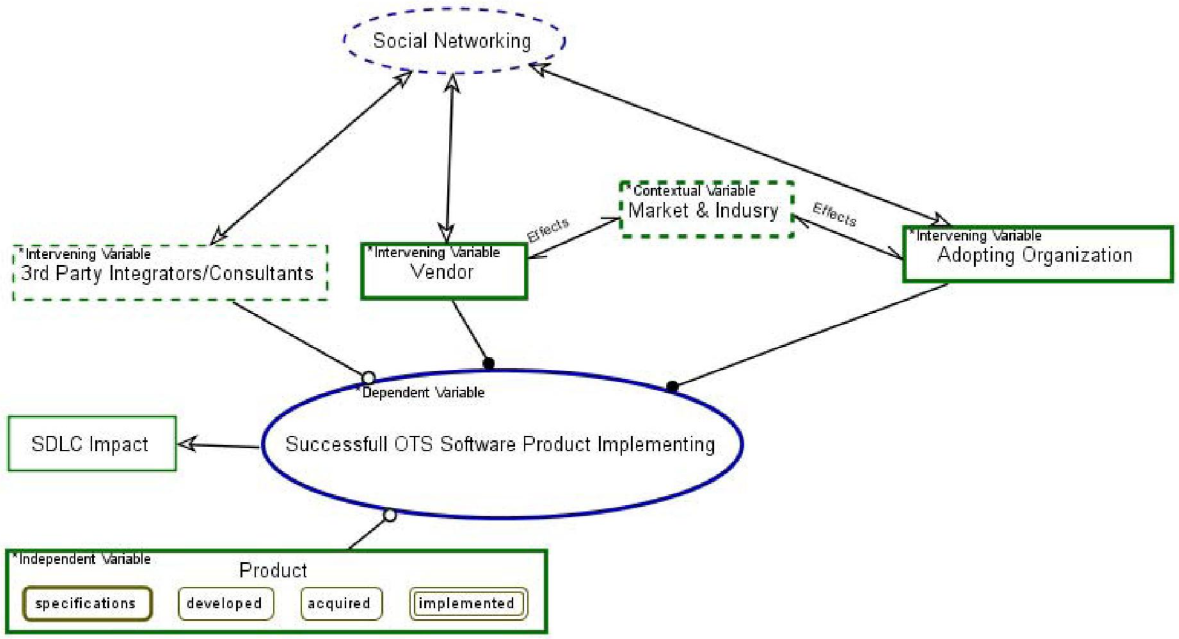

Figure 2: The larger-scale research model

Implementation Process – depicted by a blue oval – is then broken down into three sub-processes. Of these three sub-processes, the first one – the development processes – is the focus of this study.

This model, part of a larger-scale research model, may yield insights into the development process and may highlight the role of improvisation in these processes. Therefore, this research will test the OTS vendor's impact during the development process on the final product using improvisational skills.

3. Analysis: Putting all domains together: Lead Driven Software Development

While conducting the case studies mentioned above, a new approach for software development by entrepreneurial vendors has emerged. This approach – Lead-Driven Development (LDD) - emerged as a favorable approach which vendors may benefit from – see table 2.

In all 5 of 7 case studies, some form of LDD was followed. We classified the level of LDD correspondence of each vendor per each case study site on a scale of 1 to 5 – 5 as being complete correspondence. Since we are clearly examining the development process performed by the vendors and do not follow other stakeholders in this study we isolated the benefit associated with the use of LDD to be the LDD influence on OTS product sales in the corresponding case study site. Benefit was therefore observed as the influence of LDD in achieving a preliminary sale with an organization (1st sale), a higher level of benefit by achieving a post sale with that organization and the highest level - a post-post-sale – which, as we shall explain later, is a measure for successful implementation too. In addition, as the examined vendors are of entrepreneurial form, other forms of associated benefits of an efficient development process such as shorter coding times, increased flexibility of product etc. are not of much significance in these surroundings and are also almost impossible to measure.

Measuring a vendor's level of adherence to the LDD process is somewhat contrary to what we expressed before that following a certain process too formalistically, does not fit with uncertainty conditions too well. This relative paradox can be settled for now by stating that this is an improvisation-intensive formalism and hence very different from the formalism previously discussed.

Table 2 - Level of Lead-Driven Development implemented by vendors vs. Benefit in sales per vendor.

Site	LDD Level	Sale	Post Sale	PPSale
Org.1 - E	4	+	+	+
Org.2 - S	3	+	+	-
Org.3 - EI	5	+	not yet	not yet
Org.6 - Q	1	-	-	-
Org.7 - B	1	-	-	-
Org.4 - C	**2**	**+**	**+**	**not yet**
Org.5 - BM	**3**	**+**	**+**	**+**

As one may gather from Table 2, for organizations 1, 2, 3, 6 and 7, the correlation between sales, post sales and post-post-sales (i.e. vendor benefit) and the level of LDD implementation is clearly significant, positive and high. The same correlation for the second vendor's sites, i.e. organizations 4 and 5, is negative, low and non-significant.

After establishing that the use of LDD is beneficial to vendors we carefully documented this process and generalized it over the seven 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.

At the core of this model, are 12 steps as follows[1]:

Step 1: Initiation – This stage is a formal stage in the regular standard 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 [2]. Improvisation is

[1] 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 (see figure 3).

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, two actions are in place. The first is demonstrating the tool in non-profit environments and finding there a limited installation bed. This use enables on-job testing of the tool and provides the developers with important feedback on bugs, missing features and general use of the tool – a sort of a focus feedback group for the tool. The academic scenery is extremely beneficial for these situations as it also hosts great uncovered commercial potential through conventions, conferences etc. 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 [17]. Thus, in this situation we suggest vendors demonstrate the new tool buy implementing a form of improvisation at the customer's site and using the tool to perform a real work task brought in by the customer belonging to his own domain. 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 on a "What's New" page 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 improving the robustness of the tool. In addition important missing features required by the baseline customer as well as 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 this suggested format, creates overwhelming monstrous software. At this point the vendor should start to funnel out some features which cater to a small audience which by now is not part of the vendor's targeted audience. Furthermore, the vendor should try to find a connection thread guiding all other features. This decision enables further product releases each containing additional noteworthy features and bugs correction/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 Interface w/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. These tasks are highly improvisational in nature.

Step 10: Second Sale – Licenses: The first sale is by no means any indication of successful implementation or diffusion of the innovation within the adopting organization [4]. Moreover, the coupling of the HR with the OTS doesn't 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 from 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 phase of maintenance. 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 training Sessions for the users, one on one tutoring of the users and software support.

Step 12: Third Sale, Support. The earliest point one might consider as the point of successful diffusion or implantation 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 products pros and cons and knows now the true benefits associated to the product for him. Therefore, a third sale is the first true mark that the customer is truly realizing the benefits or the products and preparing for a long term use with it. Support of the Product continues now on an annual basis with milestones for new upgrades and releases as well as supplementary/complentarities.

Given that a vendor adopts this new 12 stage development process, and would like to engage in the improvisation-intensive stages the question which should arises 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 which are:

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.

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 to try out many a time risky and/or innovative solutions.

These three main factors coincide with improvisational skills factors in the literature. For example, Vera and Crossan [11] 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 to be of higher influence than others. The four factors they isolated were: expertise, teamwork skills, experimental culture and real-time information and communication.

Vera and Crossan's four factors for improvisational skills are similar to our findings. Our Teamwork skill variable corresponds to their same variable and in addition embeds their real-time information and communication variable. Our experience factor is somewhat a combination of their memory factor - which they mentioned although not finding it statistically significant - and expertise factors. Finally, the need for an experimental culture was corroborated by both studies.

Furthermore, Vera and Crossan also discussed the role of training as having a clear positive effect on improvisational skills and hence on innovation abilities.

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 sleeve solution and bypasses for many software unpredicted difficulties encountered.

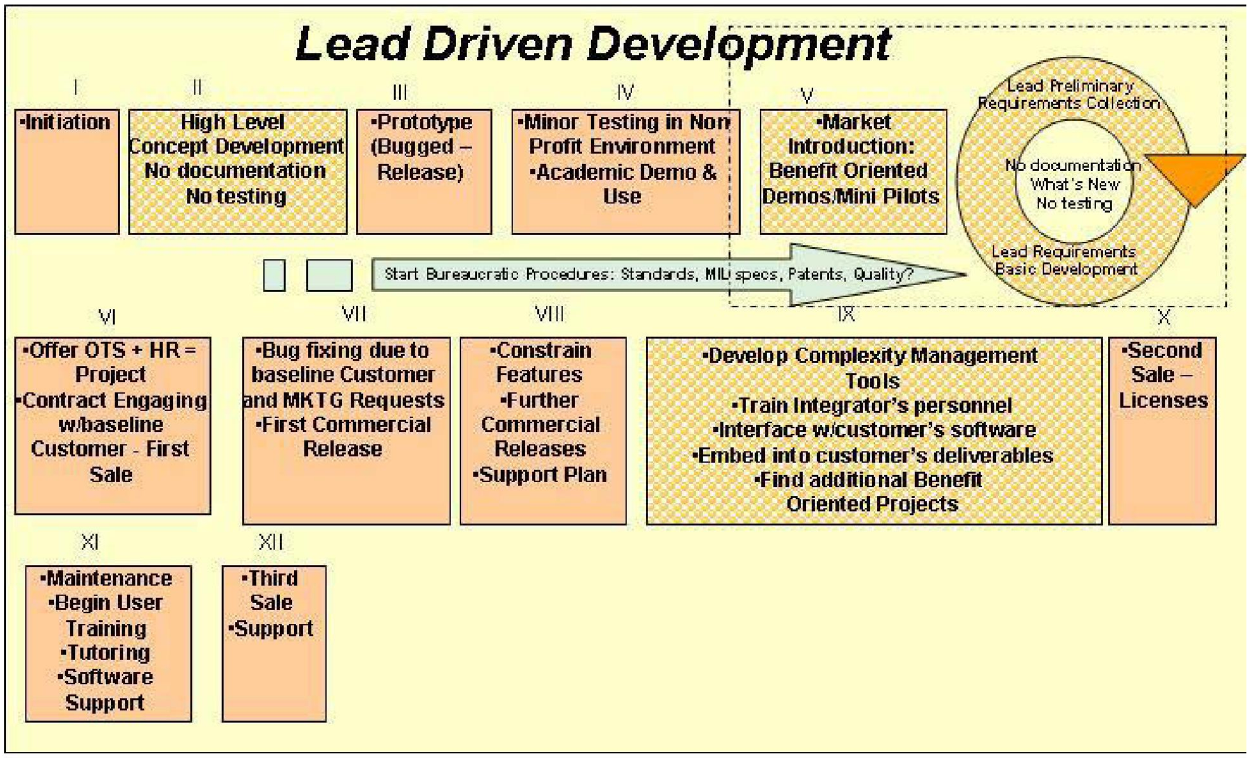
Lead Driven Development
I
•Initiation
II
High Level Concept Development No documentation No testing
III
•Prototype (Bugged – Release)
IV
•Minor Testing in Non Profit Environment •Academic Demo & Use
V
•Market Introduction: Benefit Oriented Demos/Mini Pilots
Lead Preliminary Requirements Collection
No documentation What's New No testing
Lead Requirements Basic Development
Start Bureaucratic Procedures: Standards, MIL specs, Patents, Quality?
VI
•Offer OTS + HR = Project •Contract Engaging w/baseline Customer - First Sale
VII
•Bug fixing due to baseline Customer and MKTG Requests •First Commercial Release
VIII
•Constrain Features •Further Commercial Releases •Support Plan
IX
•Develop Complexity Management Tools •Train Integrator's personnel •Interface w/customer's software •Embed into customer's deliverables •Find additional Benefit Oriented Projects
X
•Second Sale – Licenses
XI
•Maintenance •Begin User Training •Tutoring •Software Support
XII
•Third Sale •Support

Figure 3: Lead Driven development: the 12 step timeline

Teamwork skills – In general software teams, with rather high sense of collaboration usually when improvise tend to innovate more. We can specify here for the software teams the following teamwork skills: development collaboration, information sharing via email, shared drives, knowledge management portals, inner group dynamics and communication etc.

Experimental culture – 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 usually isn't backup by management should be backup by management and should also includes 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 being an industry built upon and relying upon heavily on the backbone of internet. Therefore, when improvising it is crucial to gain real-time updated information over the LAN or internet and having a variety of channels for communicating with the customer and other team members. These lines each specialize in a different type of content that may be passed: audio, video, documents, emails etc.

4. Summary

Through the observations made in seven industry case studies, regarding the successful implementations of OTS products of entrepreneurial developers, a new model for software development emerged – Lead-Driven Development. This model, highly beneficial for vendors when strictly followed, embeds action items from the IS, marketing and organizational realms. A large percentage of these action items are performed using improvisational skills.

To excel in Lead-Driven Development in general and software development improvisation in particular, entrepreneurial vendors should enhance their improvisational skills.

This study, corresponding to previous findings in the literature, describes three main factors influencing improvisational 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 [11].

Although opportunity exploitation based actions have been discussed to some extent in the literature [2], this research expands this notion from the mere founding process into the product development and software development lifecycle phases of start-ups.

Acknowledgement

The partial support of this research by the Bernard M.

Gordon Center for Systems Engineering is gratefully acknowledged.

References

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

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

[3] T. Dyba, Improvisation in small software organizations, *IEEE Software*, Volume 17, Issue 5, Sep/Oct 2000, pages 82-87.

[4] R. G. Fichman, and C. F. Kemerer, The Illusory Diffusion of Innovation: An Examination of Assimilation Gaps, *Information Systems Research*, volume 10, no. 3, 1999, pages 255-275.

[5] Franke N., von Hippel E., and Schreier M. , Finding Commercially Attractive User Innovations: A Test of Lead-User Theory*, J PROD INNOV MANAG* 2006;23:301–315.

[6] J. Highsmith, Agile Software Development: Why It's Hot!, Addison Wesley. 2002.

[7] K. M. Hmieliski, and A. Corbett, Improvisation as a framework for investigating entrepreneurial action, Academy of Management Best Conference paper 2003 ENT.

[8] C. Moorman, and A. S. Miner, "The convergence of planning and execution: improvisation in new product development" *Journal of Marketing*, 1998, vol 61, pp 1-20.

[9] F. Murray, and M. Tripsas, The Exploratory Process of Entrepreneurial Firms: The role of purposeful experimentation, *Advances in strategic management*, Volume 21, 2004, pages 45-75.
[10] R. Yin, Case Study Research: Design and Methods, Thousand Oaks, London, New Delhi: Sage, 1984.

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

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

[13] K. E. Weick, "Improvisation as a mindset for organizational analysis" *Organization Science,* 1998, vol 9, pp. 543-555.

[14] M. Crossan, M. PINA, E. Cunha, D. Vera, and O. CUNHA, "Time and organizational improvisation" *Academy of Management Review*, 30, 1, 2005, pp. 129-145.

[15] Miner et al. 2001 ""Organizational improvisation and learning : a field study" *Administrative Science Quarterly,* 46, 304-337

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

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