December 2006

Requirements Practice: What Matters to Small Businesses that Develop Packaged Software?

E. Rose
Sciences- Massey University

A. Jeffery
Sciences- Massey University

Requirements Practice: What Matters to Small Businesses that Develop Packaged Software?

E. Rose
Institute of Information and Mathematical Sciences, Massey University
Auckland, New Zealand
e.a.rose@massey.ac.nz

A. Jeffery
Institute of Information and Mathematical Sciences, Massey University
Auckland, New Zealand
observa@xtra.co.nz

ABSTRACT

This paper presents preliminary findings from a grounded theory study of managing changes to packaged software. Our aim is to develop a process theory that describes how developers resolve their main concern of managing a single product vision in a context where that product must suit diverse customer needs. The theory provides an alternate understanding of why traditional requirements engineering methods are not perceived as useful. The process theory developed here describes the interaction of the reciprocal processes of sense-making and influencing with contextual conditions and outcomes in order to explain how the participants resolve their main concern.

KEYWORDS:
Requirements practice, packaged software, grounded theory, small businesses, improvised learning.

INTRODUCTION

Packaged or "shrink-wrapped" software is software that is targeted at niche or mass, business and consumer markets. In 2003, the global market for packaged software was estimated at $US 179 billion (Software & Information Industry Association, 2004) and is predicted to continue to grow. Many firms that develop packaged software are small but much of the research on software requirements focuses on the big players such as Microsoft (Cusumano & Selby, 1997). Furthermore, software process improvement research and method designers primarily focus on custom software development, creating methods that focus on producing specifications to meet a customer's requirements, on transforming artifacts, managing dependency and coordinating a large number of developers. There has been relatively less work on small firms that produce packaged software (Fayad, Laitinen, & Ward, 2000; Sawyer, 2000; Torchiano & Morisio, 2004) despite the growth in this area and the persistence of problems such as managing changes to software products.

Field studies of practice in contexts that step outside the manufacturing/production paradigm may provide further insights and alternative explanations for the poor adoption rates of software development methods. Much of the current literature focuses on prescriptive methodologies and tool support to improve process performance and product quality. Prescriptive methods, however, do not take into account variance in development context, social process influences or individual developer capabilities. Contextual factors and social processes may have a greater influence (Guinan, Cooprider, & Faraj, 1998; Jiang & Klein, 2001) than technical factors (e.g. methods and tool support). A grounded theory approach can be used to uncover alternative ways of viewing the process of managing change to packaged software that accounts for development context. A fresh perspective can facilitate building theory with explanatory capability that has utility for practice (Glaser & Strauss, 1967; Weick, 1989, 1995).

The aim of this study is to build a process theory that 1) describes how small businesses that develop packaged software manage change requests from multiple customers and 2) that explains why they operate as they do. Process theories are useful in describing and explaining behavior in terms of the interaction of contextual conditions, human actions/interactions and consequences (Langley, 1999). Our general research question is: What is the main concern of owner-developers of small firms with respect to managing change requests for packaged software and how do they deal with it? The findings contribute to our understanding of small teams and packaged software development by examining the interactions between social structure and human action in a small New Zealand firm that develops packaged software.

BACKGROUND

In a grounded theory study, the literature review is a source of data used to evaluate the emerging theory. It is not used as a basis for constructing a theoretical framework for testing. Therefore, we provide only a brief summary of the differences between packaged and custom software development but tie it to this study in the findings and discussion sections.

Sawyer (2000) summarized the differences between packaged and custom software (see Table 1) and verified his framework using three case studies. The differences imply that issues of context and social structure may bear further study when building or extending theory (Weick, 1989, 1995). Research on managing changing requirements has primarily been targeted at custom software, large organizations and more recently at small teams within large organizations.

	Packaged Software	Custom Software
Industry	Time-to-market pressures Success measure: profit, market share, mind share	Cost pressures Success measure: user satisfaction, user acceptance, ROI
Software Development	Line positions Distant customer, less-involved Immature process Somewhat integrated design & development Design control via coordination	Staff positions Closer customer, more-involved Mature process Separate design & development Design control is via consensus-building
Cultural Milieu	Entrepreneurial, individualistic	Bureaucratic, less individualistic
Teams	Self-managed Involved in full cycle Cohesive, motivated, jelled Opportunities for large financial rewards Usually small, co-located Share a "product vision"	Matrix-managed, project-focused People assigned to multiple projects Work together as needed Salary-based Grow larger over-time, tend to disperse Rely on formal specs & documents

Table 1. Comparison of Software Development Milieus (Sawyer, 2000)

Differences in complexity/size of a project, mode of development (e.g. custom vs. packaged software), required speed of development and company size are not accounted for in requirements methods (Fayad et al., 2000). Not accounting for contextual differences may contribute to low rates of adoption. Organizations who do adopt either adopt parts of methods or adapt them to suit (Curtis, Krasner, & Iscoe, 1988; Dyba, 2001; Groves, Nickson, Reeve, Reeves, & Utting, 2000; Riemenschneider, Hardgrave, & Davis, 2002). Evidence of developer perceptions of a method as useful (in terms of reducing work effort) and compatible with current work practices can facilitate intention to adopt or not adopt (Riemenschneider et al., 2002).

The engineering paradigm behind most software development method recommendations for best practice is based on a Tayloristic model of work with an underlying ideology of positivism, functionalism and reductionism (Vickers, 1999). This may not suit the industry structure and group dynamics of small developers of packaged software. Fit with current work practices, contextual factors and the role played by human ingenuity in the development process and its outcomes, whether negative or positive, are not well supported by such methods (Dyba, 2000; Sachs, 1995). Continued growth of the packaged software market and small software development businesses, mismatch between the assumptions of existing engineering-based methods and the realities of the development environment have motivated us to use grounded theory to discover how one successful small development organization in New Zealand manages changes to a Customer Relationship Management (CRM) package.

RESEARCH DESIGN

A qualitative grounded theory research design focuses on the study of human behavior in context (Glaser & Strauss, 1967; Strauss & Corbin, 1998). An interpretive stance informed by the meta-theory, structuration theory (ST), is assumed. Structuration theory provides an analytical lens to guide examination of the interplay of social structure and human action that constitutes the process of managing changes to packaged software. A particular strength of ST is its ability to accommodate both engineered and behavioral change (Poole & De Sanctis, 2004).

Grounded theory aims to generate theoretical constructs that explain the behavior surrounding the main concern of study participants (Stern, 1980). The analytic focus is a generic social process. The aim is to formulate theory about generic social processes by grounding these processes in the data. Grounded theory uses theoretical sampling where data collection and analysis are conducted in an iterative fashion. Theoretical constructs emerge from each phase of analysis and drive further data collection. Data is collected, coded and categorized to form a large number of concepts. Higher level concepts built from sets of lower level codes are reduced in number by comparing category to category (Stern, 1980). Selective sampling of the literature and further data sampling is used as evidence to support more general categories and to look for conditions under which they co-exist in order to discover relationships.

In this study, data was collected from two owner-developers via multiple site visits. Multiple unstructured and semi-structured interviews were conducted. Email between developers and clients and from existing company documents were also used. Data were coded, compared with each other and compared with emerging theoretical constructs (Glaser, 1978; Glaser & Strauss, 1967) over a period of one and half years. Negative evidence to confront the emerging theory with alternative explanations was sought. Participants provided comments and corrections to researchers' interpretations during open coding. The emergence of theoretical concepts and relationships drove further data collection that in turn led to further theory modification. During this iterative process, memos were written to record similarities and differences in the data and to document the emergence of conceptual categories, relationships and alternative explanations of the requirements change management process.

FINDINGS

In grounded theory studies, the core category is a construct that represents the participants' main concern. Related categories are properties of the core. In the case of CRMSoft, the team's product vision emerged as the core category. It explains most of the variation in behavior noted in the data and was related to many of the other initial concepts derived from the data as shown in Figure 1. The product vision was frequently mentioned as a constraint on interaction between developers, between developer's and customers as well as on individual actions and role switching. The product vision is a malleable construct that is altered by structural conditions and human actions. Figure 1 summarizes the main relationships between the product vision and other categories found during the initial readings of the early interview transcripts.

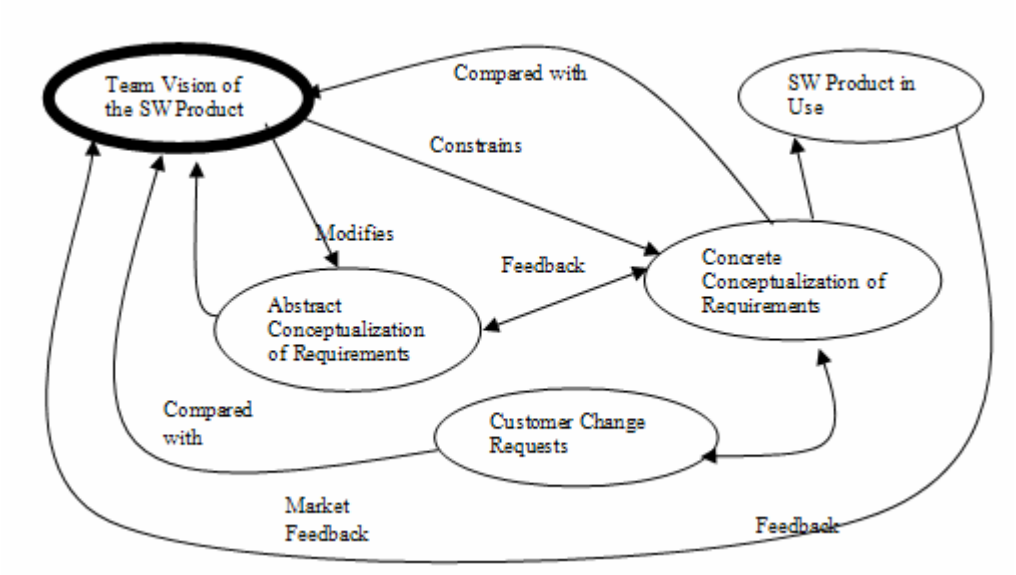

Figure 1. Conceptual Map of the Participants' View of their Requirements Practice

In order to manage the product vision, the participants engage in a process of learning about and responding to change requests. This process involves the interplay of sense-making and influencing actions/strategies. This mode of "learning in action" has been referred to in the literature as improvisation (Crossan & Sorrenti, 1997; Dyba, 2000). This type of organizational learning is unscripted and occurs while the participants are engaged in other activities. Improvisation is often associated with Jazz performance (Cunha, Cunha, & Kamoche, 1999; Hatch, 1999) and contexts of minimal social structure. Jazz players are self-disciplined, motivated, and aware of each other and their surroundings. They possess domain knowledge and technical skills acquired through extensive shared practice as illustrated below.

> *"Our relationship provides a happy medium that provides for future proofing of CRM1. We operate as two different thoughts about changes that match and merge."*
>
> *"Any changes are organization wide as it is necessary that both of us are able to share the information. For any change that is marginally complex [Shaun] and I talk it through."*

Figure 2 models the process theory that emerged from the data, replacing the initial model shown in Figure 1. A process theory describes and explains a process in terms of interaction amongst contextual conditions, actions and consequences (Orlikowski, 1993). It describes how a process emerges and unfolds but does not show cause and effect (i.e. dependent and independent variables).

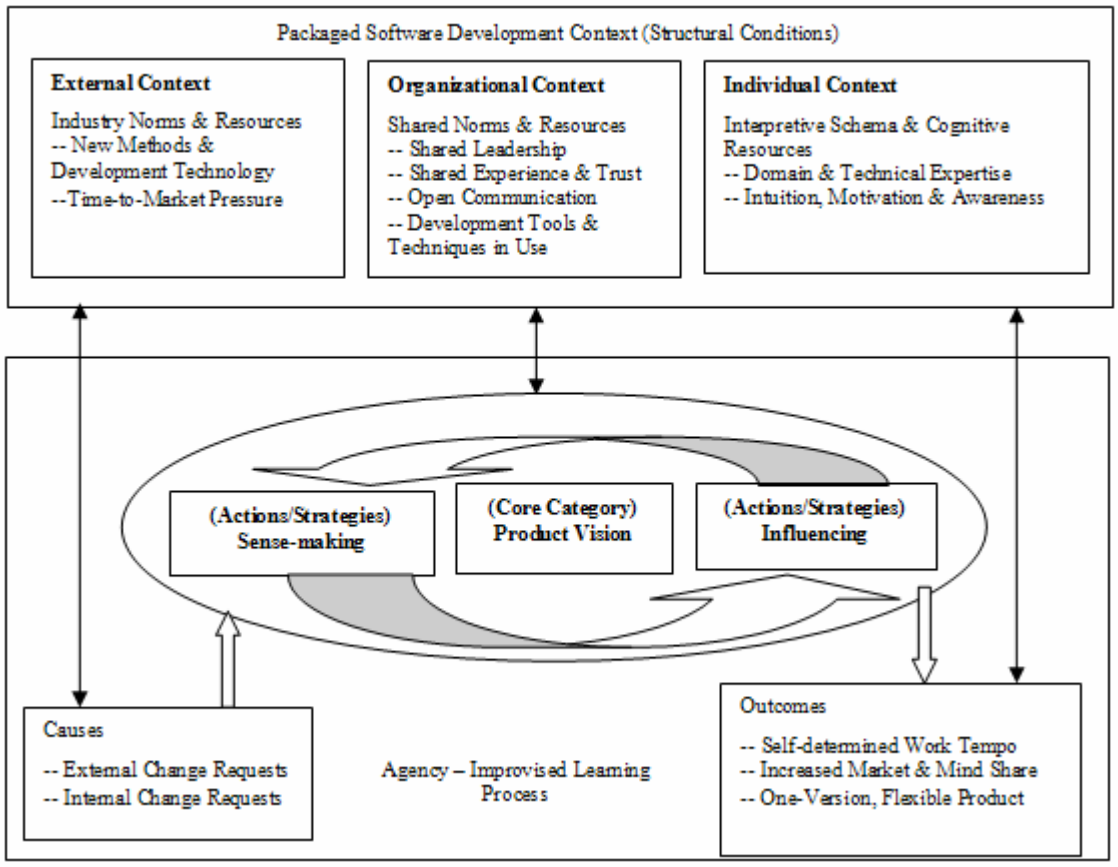

Figure 2. A Process Model of Managing Changes to Packaged Software

A change process characterized as improvised learning shapes and is shaped by the product vision, the main concern of CRMSoft. The influence of structuration theory as a theoretical coding paradigm underlies Figure 2. Evidence of influencing was found in the form of coercion strategies related to power conditions. Evidence of sense-making was reflected by communication and sanctioning strategies related to interpretive schema and norms.

> *"[Henry] and I assess whether new functions are in accordance with business direction."*

> *"A primary aim of the development direction is to avoid having multiple development directions. The most important question is whether any change will be useful to everyone, as well as being useful to the person that requested it."*
>
> *"We used real-life scenarios to illustrate issues. Whiteboards were used to describe the requirements, as well as brainstorming."*

Interpretative schema, resources and norms constitute social structure. The minimal social structure of CRMSoft is similar to that of a Jazz band. Owner/developers at CRMSoft share leadership by switching between leading (soloing) and comping (supporting) with little effort. They each have domain and technical expertise, shared experience and a vested interest as owners of the firm.

> *"We've both worked together using the same technique and I consider it to be a good fit."*
>
> *"Our actual practice is by mutual agreement – not conscious but rather based on implicit understandings."*

Contextual conditions at the industry, organization/team and individual level both facilitate and constrain the process of improvised learning, serving as both the medium and outcome of this process. Human actions/interactions/strategies constitute the complementary processes of sense-making and influencing, key properties of improvised learning. In the following paragraphs, Figure 2 is used to discuss how customer change requests are managed and why CRMSoft operates as it does. The discussion on agency deals with how customer change requests are managed. The discussion on structure describes contextual conditions that explain why CRMSoft operates as it does. The discussion on the relationships (double arrows in Figure 2) between structure and agency illustrates mutual shaping.

Structure - Contextual Conditions and Outcomes

The social structure of CRMSoft is influenced by the industry it is embedded within, the individuals, Shaun and Henry, that make-up the organization and the past experiences of these individuals.

External Context

In 2002, 86% of New Zealand businesses were classified as micro businesses (five or fewer employees). In terms of number of employees, a large number of small-sized firms and a small number of large-sized firms characterize the New Zealand software industry (Williams, 2003). Most New Zealand IT firms have 1-5 full-time employees and operate in isolation. As a result many lack capital resources and specialist business skills, and have a limited offshore presence. New Zealand's $5 billion software industry is built on a culture of innovation and creative problem solving (Gibson, 2004). The site that participated in this field study is a micro-business in the New Zealand packaged software industry which is referred to as CRMSoft to preserve its anonymity.

Successful outcomes for CRMSoft reflect the industry norms of mind share and market share but profit has less significance. Maintaining a self-determined balance between work life and home life (work tempo) is also considered an internal measure of success at CRMSoft.

> *"Any practice changes are based on a judgment about "effort and reward" – I don't think formal methods are justified unless the rewards are clearly sufficient to justify the effort required to carry out the changes."*
>
> *"I have no desire to become rich from selling [CRM1], it's a life-style choice as much as an income earner."*

CRMSoft is aware of industry resources such as development methods and recommended best practices. CRMSoft did not use a formal method to guide product development or evolution but they have adopted a number of individual recommended practices such as use of role playing/scenarios, prototyping to examine the nature and impact of potential requirements changes and basing the original product design on a modular architecture with minimal dependencies.

Organizational Context

CRMSoft is a two-person software development organization that has been in existence since 1998. It sells its products through a network of over 300 resellers and integrators and specializes in a Customer Relationship Management (CRM) product, referred to here as CRM1. CRMSoft provides solutions and services to over 80,000 customers in Australia and New Zealand. CRM1 had been on the market for two years at the time of data collection. The initial product vision originated with Shaun (all names are pseudonyms) and was based on extensive experience gained while selling CRM software. Shaun enlisted a software developer, Henry, whom he had worked with previously at a medium-sized IT business. Previous familiarity with each other's capabilities and mutual trust played a key role in their willingness to leave their current jobs and

work together under an informal arrangement, based on "less than a handshake". Leadership in CRMSoft is shared as in a Jazz performance where players switch between comping (supporting) and soloing (leading) based on the expertise needed at the time (boundary crossing).

> *"I accept that [Shaun's] role is to convey user's wishes in relation to changes and that [Shaun] has better communication skills. There is a division of skills, rather than a division of responsibility."*
>
> *"I've [Henry] not had an analyst's role, but I am familiar with dealing with changing customer requirements, and dealing with customers directly."*

The work tempo is self-monitored but influenced by the nature and volume of requests from the audience (customers). However, both have significant roles with respect to dealing with system requirements and change requests from customers. Open communication between developers and with customers is the norm.

Individual Context

The group dynamics that facilitate improvisational learning are based on the presence of individual capabilities (resources) with respect to handling change requests that rest on extensive procedural (know-how) and declarative (know-what) knowledge of the CRM application domain and technologies that support it. Shaun's extensive CRM domain knowledge was utilized to define the product vision, to differentiate it from other CRM products and to enable the use of visualization techniques such as scenarios. Currently Henry's primary role is development and Shaun often takes the lead with respect to sales, customer contact and management. Shaun and Henry were confident their scenarios were accurate because of their extensive communication and complimentary skills. This knowledge is held individually (interpretive schema) by the members of the team but is also shared via open communication. Individuals communicate their assumptions, beliefs and values to others during sense-making and influencing.

Agency – How the Interplay of Sense-making and Influencing Strategies Affect Change Management Practice

Organizational improvisation is defined as: "the conception of action as it unfolds, by an organization and/or its members, drawing on available material, cognitive, affective and social resources" (Cunha, 1999)

The initial episode of repertoire building and exploration began with Shaun's product vision of CRM1:

> *"Relatively low cost software that provides 80-90 % fit out of the box and is able to include customized modules."*
>
> *"[CRM1] is a Toyota Corolla rather than a Ferrari and it can be used out of the box by people who have never used a CRM product before."*

This episode lasted about six months and was dominated by sense-making with a primary goal (in addition to completing a marketable product) being to reach a shared understanding of the product vision and to develop shared work practices with Henry. Sense-making here is a cognitive process of understanding that involves filtering and surfacing assumptions and expectations in order to revise internal schema to make sense of a change request. During this initial episode the first code was cut, domain knowledge was shared and prototypes were used as proof of concept. There was no customer involvement. Shaun led in situations requiring domain expertise while Henry led in situations requiring technical expertise.

The first sale triggered the start of the next episode. As customers struggled with or found new uses for CRM1, change requests began to flow into CRMSoft. Some requests were incorporated as product changes; others were rejected, modified, or put off for future product releases based on evaluating potential changes in terms of effort, reward and impact on the product vision. CRM1 has sold steadily to small businesses and to business units within larger organizations for about two years. Influencing actions by CRMSoft on customer requests and sense-making processes are affected by the constraints and flexibility of the CRM1 product vision.

Recently CRM1 was sold to a large customer with the potential to expand to 200 users. The market for CRM1 is likely to grow dramatically in the near future as its visibility increases. The two owner/developers have indicated that they are concerned that growth may result in significant changes to the current tempo of work that they would find unacceptable. Henry is currently addressing these concerns by using the flexibility inherent in the product design to minimize the effort required for change. The use of technology to smooth out resource problems is preferred to adding additional developers

although they have used other developers for one-off jobs. Newcomers may not have the same mind set and would shift the current power balance.

> *"[CRM1] is based on a stable set of entities, that allow for a high degree of flexibility. Both [Shaun] and I have a high degree of agreement and understanding about the CRM market. There is little need for additional resource."*

> *"I estimate that at somewhere between 20 and 30 customers there will be a need to "formalize" the software upgrade practice. It will likely take the form of an automated method of enabling users to update their software."*

Relating Structure and Agency: Why CRMSoft Operates as it Does

Conflict is experienced when ambiguity exists in interpretive schema, when there is an inability to control resources and when disorder arises from not conforming to organizational norms with respect to shared, accepted work practices. Comfort, that is being "in the Groove" in Jazz lingo, is experienced under the opposing conditions of clarity in understanding, sharing control and order. CRMSoft seeks desired outcomes by minimizing conflict and maximizing comfort.

The domain expertise of Shaun and Henry is greater than that of their customers. This structural condition resulted in the ability of CRMSoft to formulate a strong, shared product vision and their ability to influence and make sense of customer change requests. CRMSoft has been able to realize its desired outcome strategies so far since the primary conflict has been with ambiguity with respect to the meaning and motivation behind change requests rather than with power struggles or disorder. Growth in the volume of change requests may force them to consider additional strategies to supplement limited resources in terms of human capital, but their desire to remain small and retain control has led to seeking technical solutions rather than expanding the size of the team.

The form and function of product and process changes has emerged from the interaction of sense-making and influencing actions, triggered by customer requests as they experiment with using CRM1 in different ways. Customer change requests may contain ambiguity due to differences in the interpretive schema of customers and developers. Clarity is achieved through an influencing process that involves deconstructing requests (to make sense of them) and reframing requests (using influencing strategies such as negotiation) so they meet product vision constraints in terms of balancing effort/reward for the developers while still maintaining customer satisfaction.

> *"The decision may be to implement the change as is, to modify for generic purposes, not to implement at all, or to place it on the backburner until there is sufficient customer base to both fund and use it."*

> *"The results of these discussions are then conveyed to the customer where we present ideas and question them about why they wish to have the change, suggest alternatives and drill to more detail."*

> *"My orientation is to understand how, and why, the users do what they do. The purpose is, at least, two-fold: 1) To ensure the customer is happy with the system and 2) ensure that they "blue-sky" [critically examine their current ways of doing things and, hopefully, improve on them] rather than attempt a force fit approach."*

> *"Our approach is pragmatic in that any major changes in practice that may result have to meet a test of effort and reward."*

Communications of technical knowledge, customer requests, domain requirements, and design approaches are unscripted and intertwined. Both sense-making and influencing involve communicative actions among the members of the team and with customers. Email, phone calls, whiteboards, scenario enactment, prototypes and face-to-face conversations are used to share domain and technical knowledge. The team works to continually improve the performance of the product for the audience as a whole.

> *"Our emphasis is on getting the next customer and therefore practice is aimed at maintaining an application that is attractive to more than the current customer base."*

As the audience grows and becomes more diverse, this task becomes more challenging. The bricolage aspect of improvisation comes into play due to time pressure present in the packaged software industry and the limited resources of a small firm which restrict opportunities to explore unfamiliar techniques. Founding a new business in the founders' prior area of expertise is often an indicator of a preference for bricolage, re-combing the resources at hand to design and execute something new (Baker, Miner, & Eesley, 2003).

> *"While in London I worked on a project for a bank. It was based on a brief and I worked with lead developers using a similar discussion/design process as I use with [Shaun]."*

"[CRM1] was developed in VB because I'm comfortable with it."

DISCUSSION

The results discussed in this paper are based on a single case and are therefore subject to the standard limitations of this type of research. Further work of both a qualitative and quantitative nature is needed to expand on and to test these initial results. Additional case studies conducted in a similar context using structuration theory as a lens may provide supporting findings allowing for analytic generalization of the concepts and patterns generated inductively here. In the tradition of grounded theory, this section briefly compares our findings with related findings in the literature, briefly mentions future plans for this ongoing work, avenues of research for others to pursue and outlines implications for practice. Space limitations prevent a more extensive comparison.

Newman and Noble (1990) conducted a case study in the context of custom software development by comparing four process models of user-involvement: conflict, learning, political and garbage-can finding a shift from learning to conflict to politics as the development process progressed. Our findings on influencing support the political process viewpoint (Newman & Noble, 1990) but our findings on the mixture of politics and learning as well as the improvisational nature of the learning process differ. In CRMSoft as in previous studies with a social process focus, industry, organizational and individual aspects of the context influence the way conflict (e.g. ambiguity) is or is not resolved (Gasson, 1999; Guinan et al., 1998; Jiang & Klein, 2001; Krishnan, 1998; Sawyer & Guinan, 1998). For example, at CRMSoft, the context allowed for resolution of conflict (ambiguity in requirements) through negotiation and discussion without a need to resort to political tactics (e.g. appealing to higher authority). A user or customer for CRMSoft is seeking out a CRM package and is less likely to have pre-conceived ideas about negative work impacts in the form of power losses that led to the prevalence of the conflict model found in the Newman and Noble study.

Newman & Robey (1992) found that established relationships between users and analysts persist unless critical encounters alter project trajectories (Newman & Robey, 1992). Similarly, at CRMSoft, developers see product trajectory as influenced by their past history and as relatively stable, at least up to the point of significant customer growth. Our findings on the influence of internal and external forces on the product vision and the lesser role of planning agree with previous practice-based findings that engineering methods focus on plans (accidental forces) and restrict the essential features of the development process (Carroll & Swatman, 1999).

Implications for Research and Practice

In the micro-firm studied here, product change management was characterized as an improvisational learning process constituted by the interplay between sense-making and influencing in a context of minimal structure and extensive domain expertise. The process is reactive not planned as per engineering methods. Equal emphasis was placed on maintaining the work tempo (desired by developers) and evolving the product and customer base while retaining existing customers. Practice was product-focused rather than process-focused; it emerged from shared experience and extensive expertise. The issues of process comfort and flexibility with respect to work practice and product use would benefit from further research. This ongoing study will incorporate an additional site and more formal insights from the organizational learning and improvisation literatures to further develop the process theory outlined in this paper. In particular, variances between packaged software business contexts and within businesses need to be further elaborated so the propositions of the emerging process theory can be stated more formally.

Software engineering methods and textbooks assume a contract situation where the performance requirement is to meet the customer's needs as set out in the contract. Recommended practice is to meet customer needs through greater participation of users during all stages of the process. For example, in the early design phase a significant distinction exists between packaged and custom software development. CRMSoft spent six months in design and development; and did not include any customers (users) as there were no customers at that time. Shaun's domain knowledge was the primary source of functional requirements while Henry's technical knowledge was used to establish technical constraints. The result was a product that both were satisfied with and the establishment of a working relationship that both are comfortable with. This may be difficult in environments where developers come and go.

Packaged software is developed in a situation of relatively high uncertainty (with respect to potential product success) and with a requirement to meet changing circumstances rapidly. At the same time, there is relatively long term developer continuity. The process model shows that contextual conditions such as extensive domain knowledge that exceeds that of the customer, overlapping skill sets, mutual trust and shared leadership may be important facilitators in organizations that desire a balance in work and home life while maintaining and growing mind share and market share for their product. Managers

may want to place greater emphasis on developing these abilities in individuals and encouraging these practices in the work place.

CONCLUSIONS

Using grounded theory, this study has provided insight into the reasons behind one successful packaged software development organization's requirements change management practice. These insights are represented as a theoretical framework (i.e. process theory) for conceptualizing requirements practice as improvised learning in context. Work tempo was found to be a significant process success factor, given equal weight to market and mind share in the company we studied. Starting up a new firm requires long initial hours for owner-developers which taper off in later stages as the product's market is established but periodically increase when new customers come on board with new requests. We have highlighted some major differences between practice in the studied context and theory on software development in the context assumed by engineering methods. By developing an initial process theory of the development practices of small producers of packaged software we have laid a foundation for further cumulative work.

REFERENCES

1. Baker, T., Miner, A. S. and Eesley, D. T. (2003) Improvising firms: Bricolage, account giving and improvisational competencies in the founding process, *Research Policy*, 32, 2, 255-276.
2. Carroll, J. M. and Swatman, P. A. (1999) Managing the RE process: Lessons from commercial practice. *Proceedings of the 5th International Workshop on Requirements Engineering: Foundation for Software Quality* (REFSQ '99), Heidelberg, Germany.
3. Crossan, M. and Sorrenti, M. (1997) Making sense of improvisation, *Advances in Strategic Management*, 14, 155-180.
4. Cunha, M. P., Cunha, J. V. and Kamoche, K. (1999) Organizational improvisation: What, when, where and why, *International Journal of Management Review*, 1, 3, 299-341.
5. Curtis, B., Krasner, H. and Iscoe, N. (1988) A field study of the software design process for large systems, *Communications of the ACM*, 31, 11, 1268-1287.
6. Cusumano, M. and Selby, R. (1997) How Microsoft builds software, *Communications of the ACM*, 40, 6, 53-61.
7. Dyba, T. (2000) Improvisation in small software organizations, *IEEE Software*, 17, 82-87.
8. Dyba, T. (2001) Enabling software process improvement: An investigation of the importance of organizational issues, Unpublished PhD Thesis, Norwegian University of Science and Technology, Trondheim, Norway.
9. Fayad, M. E., Laitinen, M. and Ward, R. P. (2000) Software engineering in the small, *Communications of the ACM*, 43, 3, 115-118.
10. Gasson, S. (1999) A social action model of situated information systems design, *The Data Base for Advances in Information Systems*, 30, 2, 82-97.
11. Gibson, T. (2004) Information & communication technology sector engagement strategy, Wellington, New Zealand: New Zealand Trade and Enterprise.
12. Glaser, B. G. (1978) Theoretical sensitivity, Sociology Press, Mill Valley, California.
13. Glaser, B. G. and Strauss, A. L. (1967) The discovery of grounded theory, Aldine Press, Chicago.
14. Groves, L., Nickson, R., Reeve, G., Reeves, S. and Utting, M. (2000) A survey of software development practices in the New Zealand software industry, *Proceedings of the Australian Software Engineering Conference*, Gold Coast, Queensland, Australia.
15. Guinan, P. J., Cooprider, J. G. and Faraj, S. (1998) Enabling software development team performance during requirements definition: A behavioral versus technical approach, *Information Systems Research*, 9, 2, 101-125.
16. Hatch, M. J. (1999) Exploring the empty spaces of organizing: How improvisational jazz helps re-describe organizational structure, *Organization Studies*, 20, 1, 75-100.
17. Jiang, J. J. and Klein, G. (2001) Information system success as impacted by risks and development strategies, *IEEE Transactions on Engineering Management*, 48, 1, 46-55.
18. Krishnan, M. (1998) The role of team factors in packaged software quality, *Information Technology & People*, 11, 1, 20-36.
19. Langley, A. (1999) Strategies for theorizing from process data, *Academy of Management Review*, 24, 4, 691-711.

20. Newman, M. and Noble, F. (1990) User involvement as an interaction process: A case study, *Information Systems Research*, 1, 1, 89-113.
21. Newman, M. and Robey, D. (1992) A social process model of user-analyst relationships, *MIS Quarterly*, 16, 2, 249-266.
22. Orlikowski, W. J. (1993) Case tools as organizational change: Investigating incremental and radical changes in systems development, *MIS Quarterly*, 17, 3, 309-340.
23. Poole, M. S. and De Sanctis, G. (2004) Structuration theory in information systems research: Methods and controversies, in M. E. Whitman and A. B. Woszcynski (Eds.) Handbook of Information Systems Research, Idea Publishing Group, 206-249.
24. Riemenschneider, C. K., Hardgrave, B. C. and Davis, F. D. (2002) Explaining software developer acceptance of methodologies: A comparison of five theoretical models, *IEEE Transactions on Software Engineering*, 28, 2, 1135-1145.
25. Sachs, P. (1995) Transforming work: Collaboration, learning and design, *Communications of the ACM*, 38, 9, 36-44.
26. Sawyer, S. (2000) Packaged software: Implications of the differences from custom approaches to software development, *European Journal of Information Systems*, 9, 47-58.
27. Sawyer, S. and Guinan, P. J. (1998) Software process development: Process and performance, *IBM Systems Journal*, 37, 4, 552-569.
28. Software & Information Industry Association. (2004) Packaged software industry revenue and growth, retrieved 14 December 2004 from http://www.siia.net/software/pubs/growth_software04.pdf
29. Stern, P. N. (1980) Grounded theory methodology: Its uses and processes, in B. G. Glaser (Ed.), More Grounded Theory Methodology: A Reader, Sociology Press.
30. Strauss, A. L. and Corbin, J. (1998) Basics of qualitative research: Techniques and procedures for developing grounded theory (2nd ed.), Sage Publications, Thousand Oaks, California.
31. Torchiano, M. and Morisio, M. (2004) Overlooked aspects of COTS-based development, *IEEE Software*, 21, 2, 88-93.
32. Vickers, M. H. (1999) Information technology development methodologies - Towards a non-positivist, developmental paradigm, *The Journal of Management Development*, 18, 3, 255-272.
33. Weick, K. E. (1989) Theory construction as disciplined imagination, *Academy of Management Review*, 14, 4, 516-531.
34. Weick, K. E. (1995) What theory is not, theorizing is, *Administrative Science Quarterly*, 40, 384-395.
35. Williams, J. (2003) Knowledge intensive service activities in the New Zealand software industry, Report from the New Zealand Ministry of Research, Science & Technology, Wellington, New Zealand.