Requirements in the wild: How small companies do it

Jorge Aranda, Steve Easterbrook, and Greg Wilson
Department of Computer Science
University of Toronto, Toronto, Canada
{jaranda, sme, gvwilson}@cs.toronto.edu

Abstract

Small companies form a large part of the software industry, but have mostly been overlooked by the requirements engineering research community. We know very little about the techniques these companies use to elicit and track requirements and about their contexts of operations. This paper presents preliminary results from an ongoing exploratory case study of requirements management in seven small companies, which found that (a) successful small companies exhibit a huge diversity of requirements practices that work well enough for their contexts; (b) these companies display strong cultural cohesion; (c) the principal of the company tends to retain control of the requirements processes long after other tasks have been delegated; and (d) the evidence rejects the simplistic view of a current "software crisis", as requirements errors for these companies, though problematic, are rarely catastrophic. We develop a number of hypotheses to explain these findings.

1. Introduction

Small companies form a large part of the software industry. For example, in the US, in 2002 (the most recent available figures), the overwhelming majority of software development firms, 95%, had less than 50 employees. They generated 21% of the total income and employed 28% of all employees in the area [12]. These companies are a rich source of technological innovation, and they fill a myriad niches that are either undiscovered or unprofitable for larger firms.

Despite this, the requirements engineering community has mostly overlooked their needs and characteristics. There is not one published paper in the entire history of the Requirements Engineering conferences that deals specifically with the requirements processes of small companies. This may be due to a lack of access to these companies, or to the mistaken assumptions that they are essentially no different than their larger counterparts, that they are a minor component of the economy, or that they do not pose any significant research challenges.

As a community, we know very little about the techniques that these companies use to elicit, track, and communicate their requirements, and about their contexts of operation. This paper addresses this problem by reporting on preliminary results from an ongoing exploratory case study of the requirements techniques of seven small software companies. It provides relevant details about each of the cases we analyzed, discusses the diversity of approaches to requirements engineering that we observed, and lists some implications from our findings for the requirements engineering community.

2. How do small companies do it?

We began this study in response to anecdotal evidence that requirements engineering "in the wild" differs significantly from the practices prescribed in the literature. We had observed that small software companies seem to manage their requirements in ways that bear no relation to what the textbooks say, and what is taught in undergraduate courses. Furthermore, much of the current research (e.g. on requirements modeling, specifications, traceability, etc.) seemed to be irrelevant to these companies.

In order to study this apparent mismatch, we designed a case study of small companies that would provide insights on the following questions:

- How do small companies manage their requirements? In particular, how do they elicit, document, communicate, and track them?
- How does the context of these companies affect them? Which forces shape their requirements management processes?
- Why do these companies adopt some requirements practices and reject others? Could lack of adoption be a problem of diffusion, education, or mismatching goals and

expectations?

3. Design and Execution of the Case Study

3.1. Methodology

Our research questions called for a qualitative field study of small software companies. We applied a multiple-case exploratory case study methodology [15]. The intention of an exploratory case study is to gather data with the aim of deriving specific hypotheses for further study. We established criteria for the inclusion of companies to our study, and interviewed key people at all of them. When possible, we visited their office space and examined their documentation and tools. We captured contextual details data, such as office layout and team member demographics and background. We inquired about the requirements processes of each company, and about the place of these processes within their larger project lifecycles.

Our unit of analysis was the software company as a whole, rather than individual projects or teams within them. This choice allowed us to focus on how requirements practices are affected by the culture, business context, and organizational structure of these companies. Note that a software *team* is only part of a company. We have not attempted to examine small teams within large corporations for this study.

3.2. Selection of cases

The study had a purposeful case sampling. Our inclusion criteria were the following:

- The company does software development as a primary activity.
- The company is small. To simplify selection, we chose an arbitrary threshold of companies with less than 50 employees[1].
- The company has been in operation for at least one year, which allows for some settling of its requirements practices.
- For convenience, the company must have offices in Toronto or its metropolitan area.

In recent years, Toronto has experienced a considerable development of its software industry. Many firms have formed an active hi-tech community with frequent gatherings and a high degree of openness. One of the authors of this paper is a well-known participant of this community, and this greatly facilitated the tasks of obtaining access to our cases and of reaching a healthy level of trust for our interviewees to disclose their strengths and problems.

The set of cases described in this paper is extremely diverse. This was not intentional, and it was not part of our selection criteria; rather it appears to reflect the diversity of the software industry in the Toronto area. We did not reject opportunities to include cases in our study due to similarities to other cases in our pool.

3.3. Execution of the study

In our initial contact with each candidate company, we explained that this was a case study of how small software companies manage their projects. Our contacts were generally partners or owners of their corresponding companies; when this was not the case, they were persons holding another leadership position in their organization.

We initially made contact with fourteen companies, and have conducted interviews with twelve of them. When possible, these meetings took place in each company's offices; otherwise we met in neutral environments. Interviews typically lasted between one and two hours, and we held from one to three meetings with each company. At this stage in our study, we have collected detailed data from seven of these firms. The preliminary observations we report in this paper are based on these seven.

Although the interviews had an open structure, we collected information about the following issues: organizational structure, company size, roles of key staff, line of work, types of customers, descriptions of the analysis, sales, negotiation, and development processes, communication of requirements to the rest of the team, documentation, tool support, requirements errors, and misunderstandings between the company and its customers.

4. Description of the cases

We present a brief description of each of our cases. Throughout the paper we will use pseudonyms for the companies, to protect their anonymity. A summary of some of our observations appears in Table 1, at the end of this section.

4.1. Endosymbiotic

Endosymbiotic is a small open source company that specializes in applications for the health business. It currently employs seven people, including its two founders. Its main customer, a general hospital, grants

[1] In studies of the computer industry in North America, the definition of a small software company is one with an income of less than US$23 million [13]. None of the companies we studied had an income more than 1/10th of this.

them the use of office space in its own grounds and interacts closely with them for requirements elicitation. This arrangement is highly valued by the partners, since it allows for frequent interactions with their users, paying customers, and costly medical devices.

Endosymbiotic's projects grow incrementally, and although they have an abstract set of goals, they have not been articulated in detail, since they shift depending on new technical requirements, regulations, and the strategic vision of the hospital's management.

Requirements come from three main sources: Standards, domain experts, and past experiences. It should be noted that everybody in the team worked for another company in the same business before starting Endosymbiotic. In a way, they are re-building a system that they had previously developed, and they understand many of its requirements and challenges.

The company loosely applies the Scrum [11] approach to communicate requirements to the team and assess progress: First, each project has a "product backlog" (a list of pending features, described with little detail) that acts as a sort of requirements document. Second, the team holds a "daily Scrum", a 15 minutes long, standing-up meeting, in which each team member reports on their current progress and plans. Third, a monthly demo gives visibility of the projects' status to everyone in the team and to hospital representatives, and allows for a prioritization of the features in the backlog.

Endosymbiotic's employees do not usually communicate requirements information through documentation. They use face-to-face interactions as their main means of communication. When they model their system designs, they tend to do so informally in whiteboards, without regard for proper notation syntax. Their drawings are usually photographed and posted in a website accessible to the full team.

Most people in Endosymbiotic have been working together for several years. They know one another's skill sets, and are familiarized with their loose agile methodology, which they decided to adopt as a group. Experiments in bringing new people to their team have not been successful – recently, for instance, a new employee from a different environment left the company two weeks after joining.

4.2. Agilista

Agilista follows extreme programming (XP) [2] closely, in both letter and spirit. It is a small software house formed by four people, all with an engineering background, and most of them with experience working in far larger corporations. It provides software consulting and development, mainly in the industrial automation area. Its projects and customer relations are long-term (projects often last over 2 years), and the company works on 2-3 projects at a time. Agilista is enthusiastic about its methodology and programming language of choice (Python), and rejects projects that do not fit with these choices.

Agilista's projects have no official end date or quote. The company gets paid as they go along, until the customer decides to stop the project. They do not write requirements documents. Instead, they produce a list of "stories", which can be a sentence long, and in which planning and effort estimation are based.

In the only noticeable divergence between Agilista's processes and XP, a senior consultant from the company (usually its principal) acts as the customer on-site for the rest of the team, after having interviewed people at the customer's site and acquired some domain expertise.

The company explores the requirements of their projects with throwaway prototypes that are ready at the end of the first iteration of the project (two weeks after kick-off). Agilista's principal admits that the company is happy to discard the results of this first iteration as long as it teaches them something about their project.

The list of user stories is modified and re-prioritized after each iteration, between the customer and the team members. The vague phrasing of some of the stories of a project is often clarified for the developers by the consultant acting as customer proxy, who in turn contacts the customer whenever needed. It is through these informal exchanges that requirements are communicated to the team. Furthermore, Agilista's open space layout fosters plenty of quick interactions among its employees, leading to a greater shared understanding of their status and projects.

4.3. Spark

Spark's offices, with their wide, open spaces, quirky gadgets, toys, and open kitchen, suit the image of a creative agency more than of a software company. However, their offerings are highly specialized and algorithmically complex products and web services for news corporations and publishers.

Spark currently employs 19 people, with 11 of them involved in software development. Due to the complexity of its code, it requires that several of its programmers have a graduate Computer Science degree.

The firm does not have a structured requirements management process. Feature requests are rarely documented. To define the requirements of their next release, the company's partners meet face to face with potential customers to find out their underlying needs in a sales/negotiation process that sometimes extends for a full year.

From these meetings and the consideration of Spark's own long term strategic goals, a small team at the company gives shape to the vision and features for the next release. The result of this process is not documented either. Rather, it is communicated verbally to the development team, which will make loose effort estimates based on this information.

The only routine practice the team follows is a daily standing-up status meeting, which lasts for 15 minutes, at the end of the day. Other than that, Spark's strategy may be described as follows: hire competent, creative, smart people; let them know what we want to do, and let them figure out how to do it themselves.

Spark's office space and the traits of its members suggest that the company has developed a personality that would probably reject any attempt to become structured. When we pointed out to one of the partners that this admittedly informal (if currently successful) approach might lead to trouble as his company grows, he responded that for them growth does not have a high priority: *"We're not going to become an IBM"*.

4.4. Bespoker

Bespoker is a 40-45 person software company that develops tailored enterprise applications for banks, insurance companies, and other large corporations. Of all the companies we studied, this was the one that most closely resembled typical views in the requirements literature of how requirements engineering should be done.

The partners at Bespoker appear to be convinced of the need of upfront analysis work before coding, and of the strengths of writing things down (for them, sharing knowledge is *"a matter of documentation"*). Broadly, their process is similar to the Rational Unified Process (RUP) [8], and it works as follows:

Projects start with a paid discovery phase. For small projects (4-6 months), this phase lasts a few weeks; for larger projects (up to 2 years), it may extend to 3-4 months. During this phase, one to three team leaders from the firm meet frequently with their customers to write a detailed specification, where every screen and business rule is described. This was the only company in our study that uses UML diagrams (use cases, class and sequence diagrams, statecharts) to some extent to document their specifications. For large projects, the specification consists of hundreds of pages of screens, business rules catalogues, and models.

Bespoker's partners claim that this process allows them to understand their projects clearly, and to estimate them within 10% of their actual performance.

These documents are reviewed at two levels by the customer (a business and a technical sign-off), a standard practice for contracts with corporations of the kind that the company deals with. After the sign-offs, the iterative development phase begins. To communicate the project's requirements to the team (which could be of 8-18 people), the team leaders produce a developer handbook that explains the specification and is available to the full team in a wiki for easy modification.

These documents do not deal with software design. The company does not want to have detailed design documents, claiming that *"80% of the benefit of design documentation is at the high level"*, and that at a lower level they have not found anything as efficient as simply writing code.

4.5. PhoneOffshore

PhoneOffshore is a provider of applications for mobile services with millions of subscribers, and its clients are large telecommunication corporations. The firm has between 20 and 25 employees distributed between their headquarters at Toronto and an offshore development office. Each of the company's projects takes around 6 months for 5 people – since, according to the CEO, *"less than that is boring and unprofitable, and more than that won't scale"*.

For PhoneOffshore, the requirements process is tightly woven with the negotiation process (*"sometimes they tell us what they want, sometimes we decide"*), and is largely performed to protect the company legally and to satisfy the expectations of its customers. During this stage, PhoneOffshore maintains two persons as liaisons to the customer: one technical, and one business relations. The technical liaison, who is also the project leader, talks to several groups of stakeholders to find their needs and to negotiate the requirements of the system. Some of the documents produced at this stage are legally binding, and negotiating them with the customer may take about a year for large projects. Even after the documents are signed off, the company admits that "requirements are never nailed down". In some cases, negotiating what the contract meant is a full-time job.

The project leader is also in charge of communicating the requirements to the team. He produces specifications of modules as needed, and a project plan to be used mainly as a reference by the developers. These project plans are not comprehensive, and whenever a developer picks up a work unit, he will contact the project lead to find out its implementation details. These discussions occur through email, instant messaging, and phone between the headquarters and the offshore office. The two offices do not have a considerable time zone difference, and communication usually occurs during business hours.

All of PhoneOffshore's projects are built upon a homegrown framework for mobile applications. This framework neatly modularizes every project's architecture, and the company sees it as one of its main strengths. In parallel to their normal operation, a small and trusted group at the company continues to improve its framework for the benefit of all of its projects.

4.6. Growing Web

Growing Web is a 5-person web development and consulting start-up that focuses on content management applications. They serve a wide variety of customers, and though on occasions they develop long-term projects, they also take very small projects that they can complete in as little as four hours.

This variation causes the company to have a flexible approach to process. For small projects, it is very loose. The company's owner talks to the prospect, quickly reaches an agreement, and informs his company's office manager. She, in turn, schedules the task and passes the relevant information to a developer in an email or a bug report. There are no other requirements documents produced for these projects.

For larger projects, the company follows a process resembling a waterfall model. In the sales process (the company sees this as sales, rather than requirements analysis), the owner talks to a potential customer for about an hour, and creates a rough estimate of the project's tasks and effort. This process is not paid by the customer, and it is usually done quickly. Their business model depends on the owner producing cost estimates efficiently: If they refused to work this way they could lose the project to a more flexible company.

If the prospect accepts the estimate, either the company's owner or the developer in charge of the project writes an Architecture and Design document, which is concerned essentially with the project's requirements and is read by the developers, the office manager, and the customers.

Once the document has been approved by the customer, the development phase begins. The requirements of the project are communicated to the team in a kick-off meeting, in which the owner discusses the client's needs and the list of features required. Later, when development has started, informal exchanges at their open space facilitate the clarification of requirements.

Although every project that Growing Web takes is different, most of them fall in the same domain, and the company uses its past experiences and tools to build it. In particular, almost all of their projects use a homegrown framework for web content management. In a way, the company works on the same "product" at all times, adjusting it to satisfy the needs of their current customers. When they stray from their main domain they have a steeper learning curve to go through, and riskier projects. The framework seems to have been completely assimilated in the team's heads: Whenever a client mentions that they want a website with a particular feature, the team translates that into terms that their framework handles, and even educate their clients to use the same terms.

4.7. Rentcraft

Rentcraft is a 25-people provider of rental management systems. Their team works on releases that take from 9 months to a year. It has two visible leads: a product manager, who acts as the customer liaison for the team, and a project leader, who coordinates progress.

For each release, their list of requirements considers buying criteria (features that customers want to see), usage criteria (features that customers take as default), and strategic criteria (features that allow the company to evolve the product in a direction they believe will be profitable). Initially, the list of requirements is about twice as long as what the team will be able to produce for the next release. Nevertheless, the list is passed on to the developers, who write an "Analysis and Estimation" (A&E) document, in which they informally discuss implementation alternatives for complex requirements, identify relevant uncertainties, and estimate the effort required for each feature.

When the A&E document is ready, the product manager and the project leader jointly choose the features that will be implemented in the next release, selecting only as many as it is feasible to implement given their resources. The result of this process is the "product requirements document" for the release.

For a company that sells products, such as Rentcraft, the requirements have a different emphasis than for companies that provide bespoke services. According to their product manager, *"the worst thing that can happen to us is not that nobody buys our software, but that one customer buys it"*, which would force the company to provide maintenance for it. For this reason, validation of the release's requirements is essential, and the product manager requests feedback from clients through beta tests and meetings.

We should point out that the product manager/project leader duo at Rentcraft has been working together for a long time, through several companies, and they have refined their interactions and their processes considerably. It is easy for them to know what to expect from each other. To a high degree, this same team awareness extends to most of their senior developers. For a brief period, the company had another product manager, with different processes and expectations, and he was rejected by the team. Soon afterwards, they rejoined with their old product manager, and the process to which the team was used was reinstated in full.

5. Results and observations

From the preliminary analysis of the data from this case study, we have identified four major findings. In each case, we use the findings to formulate specific hypotheses for further investigation. We discuss each of these in turn.

All the companies included in this study have requirements practices that work for them. They earn enough revenue to stay in business, and in most cases, to grow. Each has survived for a number of years in the risky world of software start-ups. These companies are led by innovative, intelligent people, who are generally knowledgeable about advanced software engineering concepts and have many years experience in the software industry.

5.1. Everyone does RE differently

When we look for common features of their requirements processes that might account for the success of these companies, we find that they all handle their requirements very differently. The diversity is striking. Each addresses the issues of elicitation and communication of requirements with different degrees of planning, structure, and documentation; and yet each considers that their choices are natural.

Several variables appear to affect the requirements practices of these companies: the type of customers, the background and skills of their developers, the preferences of their founders, the nature of their business environment, the spatial layout and geographical distance of their offices, and their number of employees. We found evidence of each of these factors at play in at least some companies. However, we do not yet have sufficient evidence about the strength of influence of each factor in the various

Table 1 - Cases summary

	Endo-symbiotic	**Agilista**	**Spark**	**Bespoker**	**Phone-Offshore**	**Growing Web**	**Rentcraft**
Company Size[1]	7	4	19	40-45	20-25	5	25
Longevity	15 months	13 years	5 years	5 years	7 years	3 years	12 years
Customers	Hospital	Manu-facturing	News agencies & publishers	Banks & corporations	Telecoms	Varied (content managemnt)	Rental companies
Type of offering[2]	Product, service	Projects	Product, service	Projects	Projects	Projects	Product
Project length /Release cycle	1 month	2 weeks	1 year	4 months – 2 years	~6 months	4 hours – 3 months	9 months – 1 year
Key requirements documents	Product backlog	Product backlog, user stories	*None*	Spec, development handbook	Statement of work, project plan	Cost worksheet, arch&design	Analysis & est., product reqs descrip.
Signs of adaptation to niche	Co-location with customer	*Insufficient data*	Year-long negotiation processes	*Insufficient data*	Homegrown framework	Homegrown framework	*Insufficient data*
Cultural Cohesion	Previous company	Engineering	CS PhDs & MScs	Previous companies	Language & country	*None*	Previous companies
Analyst	Founder	Founder	CEO/CIO	Project lead	Project lead	Founder	Product manager
Mitigation of requirements errors	Monthly demos	Iterations	Iterations	Upfront analysis, iterations	Negotiation	*None apparent*	Upfront analysis, beta testing

Notes: 1. Company sizes are approximate for cases where the company is currently recruiting and hiring new staff. 2. We categorized the company's activities according to where the requirements originate: "Projects" are custom development projects with a specific customer and limited duration, "Products" are applications intended for a wider market, and "Services" are long-term engagements (e.g web services).

cases.

Hypothesis: *The diversity of RE practices in small companies can be explained as the result of evolutionary adaptation, as these companies have adapted to a specific niche.*

According to this hypothesis, we can view the software industry as an eco-system. It would appear that, over time, differentiation occurs when companies adapt their requirement strategies to fit a particular ecological niche. Natural selection reinforces this process if companies survive in a competitive environment by being better adapted to a particular niche than others.

The implications of this hypothesis for requirements engineering research are interesting. If the hypothesis is correct, no generalized requirements technique will be suitable for all small companies. The value of any novel requirements technique will vary significantly depending on the context of the company. Ideas from RE research will not successfully transfer to small companies unless they are tailored to the particular context of the company. Just as a requirements analyst must understand the specific needs of her customers, the RE researcher must understand the varied needs of software companies: context is essence.

Further work is needed to test this hypothesis, and if it is correct, to investigate the forces that shape the evolution. It is still unclear whether it is the software company that specializes to a niche, or the niche that adapts to the company, or both. For example, companies may select requirements practices that suit their customers, or the customers may choose whom to work with based on how competing companies approach the requirements process. Our initial data indicates some support for both of these accounts.

Just because they are adapted doesn't mean there is no room for improvement. While each of the companies we studied has a requirements process that works well enough currently, these may not be robust enough to survive as the company grows, or as the business environment itself evolves.

5.2. Cultural Cohesion

A second striking observation about the companies we studied is the high degree of cultural cohesion they exhibit. In almost all cases, social characteristics shared by the group enabled it to simplify the tasks of requirements communication and coordination. Specifically, we observed cultural cohesion occurring in the following forms:

Homophily: A phenomenon that manifests as homogeneity in a social network, caused by a natural attraction of its members to similar individuals [10]. We found several instances of homophily in our cases (Agilista's shared engineering and professional background and PhoneOffshore's prevalence of employees of the same minority, among others). This homogeneity may increase the efficiency and reliability of communication, and make it easier to develop a shared understanding of the requirements.

Long term collaborations: In several cases, a notable characteristic was the long term collaboration between several of a company's members across firms. These relationships (e.g. Rentcraft's product manager / project leader duo, Endosymbiotic's full team forking from a different company, Bespoker's partners collaboration through almost two decades) enables a deeper understanding of the work style of one's colleagues, and a better estimation of their capabilities.

Rejection of radical change: For many of these companies, their current requirements practices were negotiated, agreed, and settled in the past. Newcomers that intend to change processes significantly, or that do not accept established practices, have on occasion been received with hostility and did not last long.

These phenomena lead us to suggest the following competing hypothesis to explain the diversity in the requirements processes of the companies in our study:

Hypothesis: *The choice of RE practices is irrelevant for small companies with strong cultural cohesion, as the efficiency of team dynamics overrides any benefits based on process.*

If this hypothesis is correct, it implies that the real goal of RE research should not be the creation of requirements techniques, but the study of the means through which teams acquire a shared understanding efficiently. Teams that have achieved strong cohesion do not need new requirements techniques because they have no problems achieving a shared understanding of the requirements. On the other hand, a team that lacks this cohesion might be able to overcome the problem through processes and documentation.

This hypothesis offers an alternative explanation of the diversity of RE practices than the evolutionary account given in the previous section. If culturally cohesive teams can succeed with arbitrary requirements practices, then adaptation to a business niche is no longer needed to explain the diversity we observed. The diversity occurs because, for these companies, anything works.

5.3. The CEO is the requirements engineer

For small company owners, requirements processes may well be one of the firm's most important activities: They rarely give away the role of requirements analyst to their employees. In four of our seven cases, a founder or the CEO of the firm does the requirements engineering. In the other three cases, a trusted senior figure in the company (a project leader or a product manager) takes these responsibilities.

We offer two complementary explanations for this phenomenon:

Hypothesis 1*: The skillset needed for successful requirements engineering is a subset of the skillset for successful entrepreneurship.*

In most of our cases the companies do not distinguish between the role of a "requirements engineer" and that of a "customer liaison". The person eliciting requirements is usually the face of the company, and all communications are channelled through this role. The requirements engineer, then, is often also the company's salesperson and contract negotiator, and needs skills matching these roles.

Hypothesis 2: *Requirements engineering and business strategy are inseparable for small companies.*

For a small company to commit to the development of a project implies locking a proportionally large amount of its available resources. Therefore, requirements work is also strategic management work: The decision of which projects to take or which features to include in the next release of a product is a strategic decision: it will define the company, enable it to exploit its strengths, or lead it through the risk of chartering unknown territory.

These explanations have important implications for our field, which has on occasion attempted to abstract the requirements process away from sales and strategic considerations. If this disconnect remains, it will be unlikely that owners of small companies find our proposed requirements practices applicable to their situations.

5.4. Requirements errors are not catastrophes

Every company that we talked to had stories to share about requirements errors that compromised some of their projects, and they were all aware of the importance of getting their requirements right. And yet, nobody recalled any catastrophes caused by these requirements errors.

This absence of requirements catastrophes contrasts sharply with the commonly accepted perception of a "software crisis", especially one largely caused by requirements problems. We discuss several possible explanations for this disagreement below.

Hypothesis 1: *Small companies that survive their initial phase practice* normal design*, which greatly decreases the risks associated with requirements engineering.*

As we have discussed, these companies are well established, and appear to have adapted to their business niches. An important part of this adaptation may have been a shift from a *radical design* to a *normal design* [14] approach to software development. Each company's specialization allows for the exploitation of skills and knowledge acquired previously, which could decrease the risk of software failures dramatically.

Hypothesis 2: *Small companies can fix their requirements problems more easily than large companies by virtue of being small.*

The scale at which small companies work reduces significantly the size of their projects and their coordination and communication challenges, in comparison to the large corporations which are often the subject of requirements engineering research. In small projects, it is easier to arrange for meetings with every participant and clear misunderstandings. Sharing a (sometimes open) office space also enables valuable information exchanges.

Hypothesis 3: *A single requirements catastrophe will drive a small company out of business.*

It is possible that we did not observe companies that have experienced significant requirements problems because such companies go bankrupt and disappear before we can study them.

Perhaps the most important indicator that requirements errors are not perceived as catastrophic is that they do not prompt these companies to take decisive actions to change their requirements processes. Owners recognize their issues with requirements work, but prefer to take the hits and maintain the processes that have enabled their survival so far, rather than to revolutionize their methods and risk failure.

Our evidence suggests that these companies have each arrived at a different configuration of requirements practices that is "good enough" for their contexts, and that revolutionizing their processes is risky, costly, and unpleasant, in comparison to the alternative of maintaining the inertia that has been profitable to this point. If this observation is correct, it means that small companies will never adopt requirements techniques that demand radical change.

6. Threats to validity

Construct Validity: Our main constructs are the concepts of "small software company" and "requirements management process". Regarding the first, we arbitrarily defined small companies as those with less than 50 employees. This limit is artificial, and might obscure the distinctions between a four-person and a forty-person company. Future studies will probe these differences; for now the limit allows us to explore the characteristics of that underrepresented 95% of companies in the industry.

As for the second construct, our study of requirements management processes was intentionally broad. We studied the processes through which knowledge about a project flows among its stakeholders, and therefore our construct overlaps with activities usually referred to as project management, business administration, sales, and negotiation. Although this implies a lack of precision when describing requirements processes, it enables us to investigate these processes in their natural contexts.

Internal Validity: Our major source of data for this study was the series of interviews we conducted with people at each firm. In most cases, visits to the participants' offices allowed us to corroborate some interview data. However, our reliance on interviews meant that we needed to trust each participant's description of their own processes. Participants may have omitted important facts, or we may have misinterpreted them. Observational field studies will help to uncover these problems, and we will carry them out in the short term as our case study progresses.

External Validity: All of our cases were firms headquartered in Toronto, and this might have generated geographical biases. We do not know if our findings apply to software firms around the world – and given that contextual factors seem to shape companies, the software industry of other cities might exhibit different characteristics. These differences, however, should not change the key insights from our study significantly.

Reliability: We expect that replications of our study should offer results similar to ours. Of course, the characteristics of each company under study will differ from our reports, but the underlying trends and implications should remain unchanged.

7. Related work

Small software development companies have been severely understudied by the requirements engineering community; and in instances when field studies have been reported, the size of the participating companies often goes unmentioned (as many other elements of their context), complicating the history of the field.

Lubars et al. [9] reported, in 1993, a multiple case study of 23 projects in 10 companies. They did not provide data on the size of the companies. The study focused on current practices and the problems these companies face; and they report that nearly all of the key problems were organizational rather than technical. Two years later, El Emam and Madhavji [5] surveyed 60 development projects, all using a particular systems analysis method and support tool marketed by a single company. They focused on the major challenges that these projects faced in their requirements process, and the size of the participating companies was not clear.

In 1997, Gotel and Finkelstein [6] reported on a case study of traceability by tracking the contributions structures of different participants. The case study was conducted in a 25-person "communications" company.

Carter et al. [4] presented a lightweight prototyping technique that was specifically meant to be used by small development teams (up to 12 people). Unfortunately, their emphasis was on team size, not on company size. More recently, John et al. [7] studied the use of domain analysis through a case study of a 14-person company, which was part of a larger consortium of 8 small and medium sized companies and two research institutes. The case study concentrated on how a particular domain analysis technique was used on a project within this company.

Finally, in 2005, Callele et al. [3] studied requirements engineering in the videogame industry. Their data is mainly drawn from industry reports, and it is possible that some of these come from the experiences of small companies, although that was not the focus of the study. In the same year, Alexander et al. [1] presented the results of a survey of requirements practices in several software companies. Size, again, was not in scope, but the study does provide insights regarding the diversity of approaches for requirements in the software industry, which they adjudicate to training, organizational standards, tools, first principles, and the experience of colleagues.

8. Conclusions and future work

This case study found evidence that small software companies have a number of characteristics that distinguish their requirements processes from those of large corporations. These findings challenge many of the common assumptions underlying requirements engineering research.

The findings lead us to offer the following recommendations for the requirements engineering research community:

State the context: Proposed requirements techniques may be ideal for certain contexts, and unhelpful for others. It is important to understand and to state the contexts in which a technique provides the greatest benefit to its users.

Connect RE research to business and social concerns: Requirements practices in small companies are closely tied to culture and business strategy.

Exploring these connections in RE research should lead to significant new insights.

Provide the evidence: Unless there is convincing evidence that a requirements technique is beneficial for a particular context, the corresponding companies are unlikely to risk abandoning the set of practices that have enabled them to survive, in order to obtain unproven benefits.

Provide incremental improvements: Proposals are often offered as monolithic changes in practices. Implementing them has the potential of disturbing the current set of practices greatly, which may have highly negative effects for the company. A safer approach is to offer proposals as incremental improvements that allow a firm to slowly adapt to change, facilitating the adoption of the practice and minimizing risk.

Our case study has not finished. We continue to interact with the participating companies and to incorporate more of them to our pool of subjects. We will investigate, among other things, the rationale that some of these companies have had for making radical changes in their requirements process in the past, and the effect that these changes provoked. We will assess the impact of several contextual variables in adopting a requirements technique; and we will study how strategic and negotiation considerations are incorporated into the requirements processes. Finally, we believe that we have only scratched the surface of the diversity that can be found in small software companies. We expect that a greater number of cases will shed some light into the extent of the specialization that small companies experience to survive in their complex environments.

9. Acknowledgements

We wish to thank the participants of our case study for their invaluable collaboration. Funding for this research was provided by NSERC.

10. References

[1] I. Alexander, S. Robertson, and N. Maiden. "What influences the requirements process in industry? A report on industrial practice", *Procs. of the 13th IEEE International Requirements Engineering Conference,* 2005, pp. 411-415.

[2] K. Beck, and C. Andres, *Extreme Programming Explained: Embrace Change,* Addison-Wesley, 2004.

[3] D. Callele, E. Neufeld, and K. Schneider. "Requirements engineering and the creative process in the video game industry", *Procs. of the 13th IEEE International Requirements Engineering Conference,* 2005, pp. 240-250.

[4] R.A. Carter, A.I. Anton, A. Dagnino, and L. Williams. "Evolving beyond requirements creep: a risk-based evolutionary prototyping model", *Procs. of the 5th IEEE Intl. Symposium on Requirements Engineering,* 2001, pp. 94-101.

[5] K. El Emam, and N.H. Madhavji. "A field study of requirements engineering practices in information systems development", *Procs. of the 2nd IEEE Intl. Symposium on Requirements Engineering,* Mar 1995, pp. 68-80.

[6] O. Gotel, and A. Finkelstein "Extended requirements traceability: results of an industrial case study", *Procs. of the 3rd IEEE International Symposium on Requirements Engineering,* Jan 1997, pp. 169-178.

[7] I. John, D. Muthig, P. Sody, and E. Tolzmann. "Efficient and systematic software evolution through domain analysis", *Procs. of the IEEE Joint International Conference on Requirements Engineering,* 2002, pp. 237-244.

[8] P. Kruchten, *The Rational Unified Process: An Introduction,* Addison-Wesley, 3rd ed., 2003.

[9] M. Lubars, C. Potts, and C. Richter. "A review of the state of the practice in requirements modeling", *Procs. of the IEEE International Symposium on Requirements Engineering,* Jan 1993, pp. 2-14.

[10] M. McPherson, L. Smith-Lovin, and J. Cook. "Birds of a feather: Homophily in social networks". *Annual Review of Sociology,* 27, 2001, pp. 415-444.

[11] K. Schwaber, *Agile Project Management with SCRUM,* Microsoft Press, 2004.

[12] U.S. Census Bureau. Statistics retrieved from http://www.census.gov/, Feb 2007.

[13] U.S. Small Business Administration. Information retrieved from http://www.sba.gov/, Feb 2007.

[14] W.G. Vincenti, *What Engineers Know and How They Know It,* Johns Hopkins University Press, 1993.

[15] R.K. Yin, *Case study research: Design and methods*, Sage Publications, 3rd ed., 2003.