Developers Want Requirements, but Their Project Manager Doesn't; and a Possibly Transcendent Hawthorne Effect

Daniel Isaacs and Daniel M. Berry
Cheriton School of Computer Science
University of Waterloo
Waterloo, ON, Canada
dzisaacs@yahoo.com, dberry@uwaterloo.ca

***Abstract*—This paper reports the results of a case study conducted in July 2010 of one industrial software development project to determine how the project's lack of any explicit requirements gathering process affected the project's development and the product that it produced. The study reveals that the lack of any requirements gathering process apparently led to missing functions in the product, reduced productivity among the project's members, and poor cost estimation. This lack converted a potentially profitable project into a liability. In the end, the project members completed the product, but much time was wasted. A requirements specification could have saved this time.**

Conducting the case study appears to have resulted in an increased awareness among the study's subjects, i.e., the project's manager and members, that a requirements engineering process was needed. This awareness apparently led to a Hawthorne effect, in which the project manager and members improved their requirements process. The next project conducted by the project manager was begun with an explicit requirements gathering process. This improved process continued through at least the end of July 2011, 12 months after completion of the study.

***Keywords*-company's RE process, developers, Hawthorne effect, missing requirements specifications, questionnaire**

I. Introduction

In the traditional software development lifecycle (SDLC), requirements engineering (RE) is arguably one of the most important stages. The role of RE has been described in literature as very important to identify stakeholders, detect problems, explore different solutions, and to decide what to implement [1].

Many believe that software development problems can be prevented with good RE practices [1]–[3]. However, what happens when a project goes straight to the development phase, having identified few requirements?

This paper presents the findings from a case study of one project by one organization to integrate into its suite of products a product from another organization that the first organization acquired. The goal of the study is to provide some insight on how the lack of RE in a software development project affected the quality of the developed software product. The paper reports the findings of a questionnaire filled in by the project's members. The questionnaire asked project members for their views on various RE topics relevant to the project that they were involved in.

Section II of the rest of this paper describes related empirical work addressing the importance of RE. The background of the company, its challenges in RE, the study motivation, and the case study design are described in Section III. The findings of the case study are presented in Section IV and are discussed in greater detail in Section V. The paper concludes with final thoughts about the case study in Section VI.

This paper conflates "requirements" with "requirements specification" as did the members of the project examined in the case study. They use the terms interchangeably, and "lacking requirements" means "lacking both requirements and a requirements specification".

II. Related Case Study Work

Through surveys and interviews of 76 stakeholders, including project managers, requirements analysts, customers, and quality assurers, Hofmann and Lehner [2] tried to identify the RE practices that clearly contribute to the success of a software project. They studied 15 RE teams in nine development organizations in telecommunications and banking. Among their conclusions were that

- the most successful teams expended an effort on requirements specifications that was twice that expended by the least successful teams, and
- the most successful teams did RE for a greater portion of the lifecycles of their software than did the least successful teams.

From these conclusions, Hofmann and Lehner made specific recommendations of best practices.

A full-day workshop was conducted in 2005 on the theme that upfront RE pays off in an improved development, software, or both. Each workshop presentation described a case study of the development of real-life or substantial research software in which thorough RE was done before development began [4]. A 1.5-hour summary of this workshop was presented at a panel titled "To do or not to do: If the RE payoff is so good, why aren't more companies doing it?" at the 2005 International Requirements Engineering Conference [5].

978-1-4577-1076-6/11/$26.00 ©2011 IEEE

Damian *et al.* [3], [6], [7] report, in three separate papers, the results of a 30-month, three-stage, explanatory case study, using questionnaires, interviews, and document inspection, of the RE process at the Australian Centre for Unisys Software (ACUS). During the study, ACUS was doing a concerted RE process improvement following a Capability Maturity Model (CMM) [8] assessment, and, the RE process improvement did indeed improve ACUS's software development. The studies conclude that an effective RE process from the beginning of a development improves the entire SDLC by improving the effectiveness of other processes in the SDLC, and ultimately, it improves the quality of the product developed during the SDLC.

A survey of empirical studies of RE can be found in a report by Ellis and Berry [9].

III. Case Study Description

This section describes all the details of the case study.

A. The Company

This case study of RE practices was conducted in July 2010 at the Ontario office, O, of Company X. Founded in 1981, X has around 600 employees in 26 offices around the world, and of these, from 225 to 250 are in O. X's revenues in Fiscal 2011 from about 20 product suites and about 100 bundleable services was US$99.2 million. The average software development project in O had from three to five developers and one quality assurer, took 18 ± 6 months, generated 2.8 MB of delivered source code, and followed a so-called agile [10] SDLC.

While O claims to follow an agile SDLC, it evidently did not take care to carefully follow the steps that ensured wide distribution of requirements knowledge in the absence of requirements documentation. Therefore, the authors believe that the usual SDLC at O was a very lightweight version of agile methods.

Recently, X acquired another Canadian company Y. Y's main product is PY, an interactive Web-based application. X acquired Y mainly to incorporate PY's functionality into X's own products, in order to be able to offer it to X's customers. The case study is about the project to implement PY's functionality as one of X's products, PX. X could not just use PY directly because X does not support the technology used to run PY.

O began the X project to build PX, hereinafter called "the project", in March 2008. The project was scheduled to last 18 months, but it required 24 months. The *client* for PX was a representative from the first of X's customers that agreed to beta test the new PX.

The project started with a team of 16, but by the time of the study, eight team members had quit in a span of nine months, because of job dissatisfaction, as described in the last paragraph of Section V.F. The eight remaining members, who finished the project, comprised seven developers, including the FA, and one quality assurer. The project manager is not counted as a team member, consistent with the manager's behavior that maintained a distance between him and the team. Of the eight team members, each of three was in his first job after graduating college; each of another three, including the FA, had around two years of experience; and each of the remaining 2, including the quality assurer, had more than 10 years of experience.

The FA is speaking from personal experience in describing the project. Therefore, this paper uses first person plural, i.e., "we", "us", "our", etc., to include the FA among the personnel of the project. The second author, the FA's academic supervisor, is not included among the people called "we".

B. Challenges in Requirements

One significant challenges we faced when we started the project was our lack of knowledge of PY's domain. This lack was worsened by the fact that PY's developers, project managers, and end users were geographically separated from the PX development team. Furthermore, when Y became part of X, all the PY developers, who knew everything about PY, quit rather than become X employees. Therefore, the X employees on the PX development team were entirely on their own to develop PX.

The initial stages of development can be summarized:

- X's senior management communicated to the PX project members in O that their job was to duplicate the functionality of PY.
- PY's functionality had to be migrated to a different technology, in order to incorporate the functionality into X's suite of software.
- PX's requirements were communicated by the PX project manager at O as a one-sentence requirements specification "Mimic this Webpage." while pointing to the Webpage implemented by PY. The first version of PX was not to have any more or less functionality than PY had.
- PY's functionality was not defined or documented anywhere. Information sufficient for a smooth development was not provided. As a result, the developers did not fully understand what was required to build PY.
- The implementation of PX relied heavily on each developer's own interpretation, a serious problem since each developer's interpretation was different from those of the others.

C. Study Motivation

There is evidence that RE increases productivity, improves cost estimation, reduces defects, and has many other benefits [2], [3]. However, in many industrial environments, including at O, RE is completely ignored [2], [4], [5], [11]. Consequently, the FA was inspired to conduct an empirical

978-1-4577-1076-6/11/$26.00 ©2011 IEEE

study of the impact of missing RE in a software development product. Determining this impact was the only research question considered in designing this study.

D. Design of the Case Study

This section describes the design of the case study, including the use of a questionnaire to collect the data, the questions in the questionnaire, and the reasons for the brevity of the questionnaire.

1) Data Collection Procedure: The FA invited all seven other project members to fill in a questionnaire about their attitudes towards RE in the project. These seven did not include the project manager, because, as explained above, he was not really considered part of the team. All seven project members returned filled in questionnaires. The questionnaire was an adaptation of that used by Damian *et al.* [7].

2) The Questionnaire: The complete questionnaire is found in the Appendix. Many a question of the questionnaire is answered with a 5-point Likert scale that is given directly below the question. If a question with a Likert scale is ended with a "Why?" or if a question has no Likert scale, then an open-ended textual answer is expected.[1]

The questionnaire was designed to be short and easy to answer in order to encourage more and complete responses from busy employees who were behind schedule in their current project to develop PX. The fact that all of the project members (other than the FA who could not be expected to fill in his own questionnaire) responded and responded fully says that this decision was not wrong. In any case, the respondent were the FA's co-workers in the project. Therefore, the FA could and did ask his co-workers follow-up questions to clarify their answers and to understand their real meanings.

E. Limitation of Method

The FA's presence on the project team opens the possibility of personal bias in the reporting. The FA has tried to mitigate this bias by the use of a questionnaire of the other seven project members to gather information from which his conclusions are drawn. Nevertheless, the FA did the data analysis and drew causality conclusions based on the analysis. On the other hand, the second author, with his more than 45 years of experience in software development, agrees with the results of the analysis and the causality conclusions.

The reader should remember that this paper reports a *case study*, which is a description of one company's experience with one of its projects. The conclusions of this paper, while appearing to be valid for the experience, cannot be generalized beyond that. The authors hope that others with similar experiences will report them so that generalization can occur from repeated occurrences of similar experiences.

[1]One could argue that the Likert scale of Q4 should have been used for Q1. Nevertheless, discussions between the FA and the respondents make it clear that each respondent interpreted the scale of Q1 as that of Q4.

IV. Case Study Findings

A. General Feedback about the Lack of Requirements

Answers to Q1 showed that even though requirements were largely missing in the project, all seven project members thought that requirements were at least important, while five of these seven, thought that requirements are far more important.

Answers to Q2 showed that the lack of requirements influenced their work in many ways. In project members' own words: "It would have been easier to put all of the components together.", "... too much guessing regarding expected results", "not enough time to develop good test cases, which made testing not being complete enough", "It hampered my work creating unexpected results.", and "We had to go back and rewrite sections of the code, causing project timelines to increase."

Answers to Q3 showed that six of the seven project members thought that they should have spent more time on the requirements phase, with two of these six saying "Far More" instead of just "More". In these project members' own words: "RE provides a better understanding of what exactly needs to be done.", "RE gives a crystal clear understanding of what the user is looking for and what he/she wants to achieve from this project.", and "RE would have prevented many different problems and miscommunications with management." Only one developer of the seven project members thought that he should have spent far less time on the requirements phase. (See the last paragraph of Section V.A.)

B. Definition of the Problem, and the Possible Benefits from Understanding the Problem

Q4 examines the four phases of the project's SDLC: designing, coding, testing, and documentation. For each phase, each respondent had to choose one of five possible degrees of importance, from "Very Important" through "Not Important At All". Figure 1 shows the number of responses per degree of importance for each phase. In this chart, each degree of importance is represented by a bar with a different pattern. For the designing phase, six of the project members thought that understanding the problem is very important, while one agreed that it is important. For the coding phase, all of the project members agreed that understanding the problem is very important. For the testing phase, three project members thought that understanding the problem is important, while another three, including the tester, thought that it is very important. Only one project member thought that understanding the problem is not important at all for the testing phase. For the documentation phase, each answer had at least one vote, with each of "Indeterminate" and "Very Important" having two votes.

Answers to Q5 show that six out of seven project members thought the amount of rework done in the current project was

978-1-4577-1076-6/11/$26.00 ©2011 IEEE

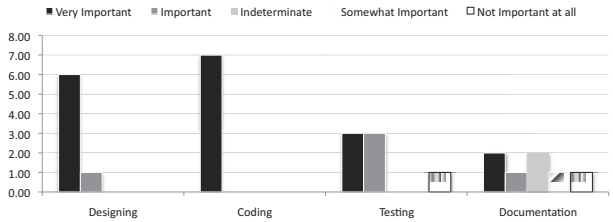

Figure 1. Importance to Different Phases of the SDLC of Understanding Requirements

about the same or more than in previous projects, in which they did not do any RE, as well. Three of these six project members thought that there was more work in the current project. Only one project member, a developer, thought that there was far less rework than in previous projects.

In their answers to Q6, regarding the effect of the lack of requirements on productivity, project members said: "Productivity was negatively affected by the amount of work that had to be redone.", "Lack of RE negatively impacted productivity.", and "Lack of information always results in unexpected results." Regarding the lack-of-requirements effect on product quality, project members said: "It deteriorated product quality because you have to keep changing the design and code as the requirements change." and "It reduced the initial product quality, however the later product is still of good quality, but it took much longer to get to this point due to the lack of proper requirements."

C. Estimating Effort, Cost, and Development Time

Answers to Q7 show that the lack of requirements affected the effort, cost, and development time estimation process negatively. In particular, five of the seven project members strongly agreed that the lack of requirements affected their estimation, while two agreed that the lack made a difference.

In response to Q8, about discrepancies in estimations for the design phase, one project member said: "The design had to be reworked several times due to new discoveries and additional requirements." For discrepancies in implementation, one project member said that: "The implementation had to be restarted due to the changes in design." For testing, one project member said: "It was usually a set back since there was no proper understanding of what the system was required to do." For the documentation phase, one project member said that: "It was very late as it had to be started late in the project."

In response to Q9, one project member said that in order for the developers to improve their estimations, they would need to "have a list of requirements and a design document about how to code the project." Others said: "More clear guidelines and expectations would have helped, and additional overview of the work that was needed to be done.", and "With additional time, I would have known of various functional requirements before the development had begun."

V. Discussion

This section draws conclusions from the data.

A. General Feedback about the Lack of Requirements

The evidence in this study indicates that the the developers and the quality assurer in the project believe in requirements. Most developers agreed that requirements are far more important, really meaning "very important", as explained in Footnote 1. In fact, all project members agreed that RE improves understanding of details, dependencies and complexities. In spite of this agreement, there was no real requirements phase in the project. The authors believe that the project manager skipped the requirements phase because of the pressure from senior management to be the first on the market.

Based on the FA's experiences at O and on informal discussions with his fellow employees, the FA concluded that while the project developers liked requirement specifications, the project's manager did not. The FA drew this conclusion from the manager's behavior. The manager seems to have resisted any suggestion of sitting down with the entire development team to figure out requirements collectively. Instead, he approached individuals and asked each to build a prototype of one feature, to interact with the client until the feature was implemented correctly, and finally to report back to only him with the completed prototype. This way of working was agile in the sense of continuous interaction with the client, but not the sense of communication with all in the team.

The FA believes that the project manager associated knowledge with power and job stability. The project manager seemed to believe that if he were the only one that knows an important something, he would be needed at all times and be indispensable. A requirements specification would expose this knowledge early and make it available to everyone in the project team. Thus, a requirements specification would be very low on the list of the project manager's desires. Ironically, because there was no systematic, coordinated attempt to figure out all the requirements up front, the manager probably knew no more about the requirements than did the project team collectively.

The absence of well defined requirements almost certainly led to the errors that occurred during the development. The resulting rewriting wasted time.

For Q1 and Q2, there was a risk that a project member would misstate his thoughts because he desired to give an answer that will be perceived as correct. In order to mitigate this risk, Q3 was designed as a corroborating question in order to see how requirements affected each project member's work. Additionally, Q3 elicited discussion about

978-1-4577-1076-6/11/$26.00 ©2011 IEEE

problems that arose in the project and how RE would have helped.

As mentioned, one developer thought he would have spent less time in the requirements phase. The relatively junior developer offered as a reason: "I would still need to spend a lot of time in the designing phase." After the FA talked with the developer, it was clear to the FA that the developer was talking about the phase occurring after requirements specification but before coding. This developer was on his first job after graduation from college. As seems to be typical with junior developers, this developer seems to value more discovering the variables, properties, and methods of the different objects that PX needs than taking the time to understand what is required of PX.

B. Definition of the Problem, and the Possible Benefits from Understanding the Problem

When asked for the importance of understanding features and requirements in the different phases of the SDLC, the project members seemed to believe that understanding features and requirements was relevant for most phases but not for documentation. Based on what the FA knows about the project there are several possible explanations. One is that developers were not forced to document their work. Thus, they did not value the documentation phase at all. Another is that documentation was done only after development. Therefore, understanding features and requirements did not seem important to the documentation phase. This conclusion is strange given that a requirements specification, if produced, would inform more than just requirements. Furthermore, a requirements specification could be used as a basis for test cases and the user's manual. Nonetheless, understanding what to do on design and coding was recognized as extremely relevant since with well understood requirements, rework is lessened, the effectiveness of communication increases, and developers are able to make better decisions.

In fact, the amount of rework on this project was about the same as on other O projects, revealing that developing software without understanding its features and requirements was the norm at O. The project manager had apparently not learned from previous experiences, thus strengthening the FA's conclusion that the project manager did not like requirements specifications.

It can be said that the PX development team was actually quite productive when measured purely by lines of code produced per hour, i.e., gross productivity. Nonetheless, at times, the development team produced the wrong results, making its net productivity considerable lower than its gross productivity, as thrown out lines of code are subtracted from the numerator, and the rework time is added to the denominator.

However, the work was not well distributed. Some developers were the sole owners of their components. In addition, the development team had no meetings to propagate knowledge. Consequently, when a project member went on vacation or on a sick day, productivity was seriously hurt.

Initially, organized RE did not happen. Consequently, the development team did not understand the product. The developers and testers were determining requirements on the fly as they were coding or writing test cases, and each was deciding somewhat independently. As a result, the product quality was initially quite poor. Over time, the development team was able to gather enough knowledge to develop the product. Consequently, the overall product quality was acceptable in the end. Nonetheless, the product was finished later than expected and later than required.

C. Estimating Effort, Cost, and Development Time

The development team clearly indicated that they believed that having requirements can improve software development and project management estimations. When asked why discrepancies might exist between estimations and actual, project members' opinions were very similar. The most commonly stated reason was that a lot of work had to be redone, due to missing specifications. When asked how estimations could have been improved, the project members' opinions were again very similar. The team needed a better understanding of the requirements and their scope.

D. The Middle Stages

Several problems happened because of the lack of requirements and communication between the project manager and the developers.

The Quality Assurance (QA) team started to test the product. They compared the old product against our newly created product. If the new system was missing some functionality or behaved different from the old system, the QA team opened a ticket in the bug-tracking application, BTA. However, the interaction between the QA team and the development team had some issues. Only one member from the QA team was available for working closely with the project. Historically, O does not invest too many resources in testing any of its developed software. Furthermore, after the development team spent nine months, October 2008–July 2009, sharing knowledge with one member from the QA team, he was moved to another project. Consequently, the development team had to start from scratch with a new member from the QA team. Therefore, testing was delayed until the new member understood the product. By the end of June 2010, the QA team had logged 681 tickets. The developers plugged away at each of these tickets as a bug to fix.

The FA, for the purposes of this study, decided to examine these tickets to determine their true origins. After examining the first 100 of these tickets, he stopped. Already, 37 of them were the result of missing requirements, and it was clear that a similar fraction of the 681 would be the result of

978-1-4577-1076-6/11/$26.00 ©2011 IEEE

missing requirements. The remaining 63 of the tickets that he examined were indeed bugs inserted during programming of a known requirement.

The senior management team that approves X's company-wide development of new products had a hard time approving this project for continuation. The main reason was that they thought the application was not robust enough because of all the tickets that were logged in BTA.

There were other issues such as: several developers' having spent a month working on some functionality, when another developer said: "You should use these libraries." Suddenly, the one-month struggle was over in five minutes. Also, sometimes, when developers fixed one problem, other functionalities of the application stopped working.

E. The Positives

There were some positive aspects to the project even though it began with an understanding of few to no requirements. For this project, we had access to the old system that we had to replicate. Consequently, we could see how the old user interface looks and duplicate it, with our new components. Therefore, we did not spend any time trying to come up with an interface, thus saving some development time.

The old system's relational database is IBM's DB2. In the beginning stages, this fact was thought to be a problem, because no one in the team had DB2 expertise. However, knowing DB2 proved not to be necessary, since the old system wrote the queries into a log file. Therefore, we had PX write out the same queries to the same file, and we had the log file given directly to the database so that it would perform the required actions such as edit, save, and submit.

F. The Final Stages

In the final stages of the development, even though the developers were able to play with the old system, requirements were still missing. For example, if some fields were filled, other fields were automatically marked as required. The forms were so complex that this marking was not easy to notice. Consequently, a lot of tickets were opened about these kinds of issues.

After working for two years on the same Webpages, developers started to miss details. Usually, other users, new developers, or the client saw details that people involved in the project for years could not see. It was embarrassing to us when the client noticed the new product lacked a basic functionality the old product had.

Near the end of the project, we tried to make things right. We tried to follow more of a truly agile SDLC [10]. Every Monday, we reviewed a list of bugs and missing requirements to choose those that we would try to fix by Friday. At the same time, we reviewed a list of requirements from the client's users to choose ones that were feasible to incorporate by Friday. Finally, on Friday, we deployed the changes completed since Monday. The QA team tested the new PX and communicated their findings to the project manager and the development team. On Monday, we started all over.

Although the project started to move at a faster pace, at this point it was too late. The project's manager lost credibility with senior management. In fact, one day, the project's manager announced that the project had changed from something profitable to a liability. Project members also suffered, as no one got salary raises and no one was promoted to higher positions, raises and promotions being the key rewards within X for an employee's having done well.

With all this adversity and stress, several developers quit the company. The developers were doing all they could to produce results, but they felt that their work was not appreciated. The questionnaire responses show that the developers felt that had there been a good RE process in place, the team would have achieved its goals on time. In that circumstance, perhaps most of the team would still be working for X; at the very least, the FA would still be there. Thus, the lack of requirements affected not only project timelines and customers but also the personal lives of team members.

G. The Road to Improvement

How could we have improved the course of this project? Hoffman and Lehner [2] state that there are three factors that contribute to project success: knowledge, resources, and process.

It is important to have the experience and expertise on a team to achieve effectiveness. However, there's always a dilemma with knowledge. Hoffman and Lehner identified the thin spread of application domain knowledge as one of the most salient problems in software projects [2]. As mentioned and explained above, the PX project manager appears to have associated knowledge with power and job stability. He appears to have believed that if he were the only one that knows everything, he would be indispensable. Moreover, the manager apparently mistakenly believed that no other employee had this special knowledge that he had. After eighteen months, the project manager gathered the development team to explain the concepts he thought needed to be understood to complete the project. However, by this point, most of the team had figured out a way to understand the concepts, which made the meeting meaningless. Such a meeting would have been invaluable much earlier into the project and would have saved a lot of time, tickets, and grief.

Guinan, Cooprider, and Faraj [12] report an average RE team size of 5.6 members in the 66 projects they studied. It is safe to say that the more people allocated to well-defined RE, the higher are the chances for the project to succeed. In our case, the team was small. Consequently, it is unrealistic to have five or six people devoted to only RE. However, if the project manager had had all project members thoroughly

978-1-4577-1076-6/11/$26.00 ©2011 IEEE

exercise the legacy product prior to starting development, the entire team would have understood what the new product had to do. In this case, 100% of the team would do RE, and then it would do implementation based on its own understanding of the requirements.

Our RE process got to the point at which project members could account for stakeholders' learning curves and for any requirements negotiation. Also, our architecture was able to support most of the changes that the client requested, even if the requirements were always changing. In fact, project members easily expanded the forms to comply with recent changes and new standards that the Canadian and United States governments had instituted for software in PX's domain.

H. Hawthorne Effect: An Impact on Future Projects

The conduct of the survey had an unexpected and surprising Hawthorne effect. "The Hawthorne effect is a form of reactivity whereby subjects improve or modify an aspect of their behavior being experimentally measured simply in response to the fact that they are being studied ..." [13].

About a month after completion of the survey for this study in July 2010, the project manager in charge of PX was given the responsibility to lead a new project for a new client. As of September 2010, this new project was still in its early stages. Nonetheless, the FA already noticed changes in project members and the manager. Conducting the case study among the project members appears to have resulted in awareness that changes in the RE practices were needed. In a project member's own words: "Moving forward, we need clearly defined requirements since they would have saved more time and efforts in terms of reducing the back and forth communication in the development and testing stage."

This new project presented an opportunity for project members and the manager to do things differently. The project manager was taking full advantage of the opportunity. In this new project, the client has been required to write a business requirements document which describes: business requirements, business policies and regulations, nonfunctional requirements, business data models, and much more. Knowledge was not withheld anymore, as the business requirements document is frequently updated and distributed among developers. Additionally, developers were invited to listen to telephone conferences that the project manager had with the client. The new client was expected to fully cooperate with the fields and search criteria that will be shown in the user interface. At times, it was hard to believe that this was the same project manager that developed PX. Evidently, the questionnaire made project members aware of RE, and they in turn made also the project manager aware of RE. Thus, the FA's case study seems to have induced a sustained after-the-fact Hawthorne effect.

As Julio Leite asked in private communication, "If the developers had learned about the importance of RE in their software engineering education, why did they need the questionnaire to motivate them to improve their RE?" Perhaps, answering the questionnaire during a challenged project was needed to make the book-and-lecture learning relevant to real life.

As pointed out by Daniel Schwabe in private communication, there are cases of clinical practice in which the Hawthorne effect has been used *intentionally* to motivate patients' sustained compliance with recommended health-improvement practices, e.g., for oral hygiene [14].

The FA has checked with his former teammates on the PX project and has determined that as of the date of this writing in the end of July 2011, some twelve months after the conclusion of the study, the improved RE process continues. One of the former teammates that the FA asked said: "We are trying to deliver a product but the client kept changing it ... but it [the process] *is* still improved." So far, there appear to be fewer defects, but the "product is still not finished so it's hard to tell." We are still "on track [to complete by the deadline] because of some padding we have been giving ourselves. We are doing a better job on this one." Finally, the customer seems to be satisfied. "They have been pretty happy with what they've been seeing so far."

The one junior developer who had reported that he would have spent less time in the requirements phase approached the FA to ask if he could change his responses to the questionnaire! Even he had gotten the message of the importance of requirements.

Perhaps, a good way to get an organization to improve its RE process, particularly when it has none is to conduct a survey like that used for this case study in the middle of a challenged project, when it is apparent to *all* how challenged it is, even if no one admits out loud that it is challenged!

VI. Conclusions

This paper reports the results of a case study that investigated how the lack of requirements affected one software development process. Data from the study show that the lack of requirements understanding and a requirements specification had a direct impact on design, coding, testing, cost estimation, and project management.

That a product *can* be developed with little or no RE is deducible from the fact that we were able to develop the product without any serious RE. Nonetheless, the road was very painful. Ignoring RE made answering questions such as: "When will it be finished?" or "Will it meet the customers' expectations?" hard to answer.

Getting industry to regularly do a complete job of RE for its developments is a cultural change that will take years. Convincing senior and project managers the value of RE is not an easy task. Perhaps, just getting people to vocalize the unmentioned, but clear difficulties they face and making people aware of other possibilities can help speed up the cultural change.

978-1-4577-1076-6/11/$26.00 ©2011 IEEE

ACKNOWLEDGMENTS

The authors thank Daniela Damian, Julio Leite, Daniel Schwabe, and the anonymous reviewers of earlier drafts of this paper for their comments. Daniel Berry's work was supported in parts by a Canadian NSERC grant NSERC-RGPIN227055-00 and by a Canadian NSERC–Scotia Bank Industrial Research Chair NSERC-IRCPJ365473-05.

REFERENCES

[1] B. Nuseibeh and S. Easterbrook, "Requirements engineering: A roadmap," in *Proceedings of the Conference on The Future of Software Engineering*, ser. ICSE '00, 2000, pp. 35–46.

[2] H. F. Hofmann and F. Lehner, "Requirements engineering as a success factor in software projects," *IEEE Software*, vol. 18, no. 4, pp. 58–66, 2001.

[3] D. Damian and J. Chisan, "An empirical study of the complex relationships between requirements engineering processes and other processes that lead to payoffs in productivity, quality, and risk management," *IEEE Transactions on Software Engineering*, vol. 32, no. 7, pp. 433–453, 2006.

[4] requirements-engineering.org, "RE Day 2005 Website," 2005, http://www.requirements-engineering.org/REday05/.

[5] D. M. Berry, D. Damian, A. Finkelstein, D. Gause, R. Hall, and A. Wassyng, "To do or not to do: If the requirements engineering payoff is so good, why aren't more companies doing it?" in *Proceedings of the IEEE International Conference on Requirements Engineering (RE)*, 2005, p. 447, http://www.requirements-engineering.org/REnotDonePanelRE05/.

[6] D. Damian, D. Zowghi, L. Vaidyanathasamy, and Y. Pal, "An industrial case study of immediate benefits of requirements engineering process improvement at the australian center for unisys software," *Empirical Software Engineering*, vol. 9, no. 1-2, pp. 45–75, 2004.

[7] D. Damian, J. Chisan, L. Vaidyanathasamy, and Y. Pal, "Requirements engineering and downstream software development: Findings from a case study," *Empirical Software Engineering*, vol. 10, no. 3, pp. 255–283, 2005.

[8] M. C. Paulk, B. Curtis, M. B. Chrissis, and C. V. Weber, "Capability maturity model, version 1.1," *IEEE Software*, vol. 10, no. 4, pp. 18–27, 1993.

[9] K. Ellis and D. M. Berry, "Quantifying the impact of requirements definition and management process maturity on project outcome in business application development," School of Computer Science, University of Waterloo, Waterloo, ON, Canada, Tech. Rep., 2011, http://se.uwaterloo.ca/~dberry/FTP_SITE/tech.reports/EllisBerry.pdf.

[10] Agile Alliance, "Principles: The agile alliance," 2001, http://www.agilealliance.org/.

[11] P. Morris, M. Masera, and M. Wilikens, "Requirements engineering and industrial uptake," in *Proceedings of the Third International Conference on Requirements Engineering (ICRE'98)*, 1998, pp. 130–137.

[12] P. J. Guinan, J. G. Cooprider, and S. Faraj, "Enabling software development team performance during requirements definition: A behavioral versus technical approach," *Information Systems Research*, vol. 9, no. 2, pp. 101–125, 1998.

[13] Wikipedia, "Hawthorne effect," Viewed 25 May 2011, http://en.wikipedia.org/wiki/Hawthorne_effect.

[14] P. H. Feil, J. S. Grauer, C. C. Gadbury-Amyot, K. Kula, and M. D. McCunniff, "Intentional use of the hawthorne effect to improve oral hygiene compliance in orthodontic patients," *Journal of Dental Education*, vol. 66, no. 10, pp. 1129–1135, 2002.

APPENDIX: THE QUESTIONNAIRE

Section A: General Feedback about the Lack of Requirements

Q1. How important do you feel requirements are?

A: Far More	B: More	C: About the Same	D: Less	E: Far Less

Q2. How do you feel that the lack of requirements influenced your work?

Q3. Based on your experience on this project, would you spend more or less time in the requirements phase of the development? Why?

A: Far More	B: More	C: About the Same	D: Less	E: Far Less

Section B: Definition of the problem, and the possible benefits from understanding the problem

Q4. In your design, coding, testing, or documentation activities, how important was it to understand the features and technical requirements?

	Very Important	Important	Indeterminate	Somewhat Important	Not Important At All
Design					
Coding					
Testing					
Documentation					

Q5. In contrast to previous experiences, has there been more or less rework during development (but before deployment):

A: Far More	B: More	C: About the Same	D: Less	E: Far Less

Q6. How do you believe the lack of requirements improved or deteriorated: (a) productivity and (b) product quality.

Section C: Estimation

Q7. Did the lack of requirements affect your cost estimation process?

A: Strongly Agree	B: Agree	C: No Effect	D: Disagree	E: Strongly Disagree

Q8. When thinking about your estimates, if any, can you think of reasons for discrepancies? With respect to (a) Design, (b) Implementation, (c) Testing and (d) Documentation.

Q9. How could you have improved your estimations?

978-1-4577-1076-6/11/$26.00 ©2011 IEEE