Success factors in managing legacy system evolution

A case study

Huijgens, Hennie; Van Deursen, Arie; Van Solingen, Rini

DOI
10.1145/2904354.2904363

Publication date
2016

Document Version
Accepted author manuscript

Published in
Proceedings - International Conference on Software and System Process, ICSSP 2016

Delft University of Technology
Software Engineering Research Group
Technical Report Series

Success Factors in Managing Legacy System Evolution: A Case Study

Hennie Huijgens, Arie van Deursen and Rini van Solingen

Report TUD-SERG-2016-009

TU Delft

SERG

TUD-SERG-2016-009

Published, produced and distributed by:

Software Engineering Research Group
Department of Software Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4
2628 CD Delft
The Netherlands

ISSN 1872-5392

Software Engineering Research Group Technical Reports:
http://www.se.ewi.tudelft.nl/techreports/

For more information about the Software Engineering Research Group:
http://www.se.ewi.tudelft.nl/

Note: Accepted for publication in the proceedings of the International Conference on Software and System Processes (ICSSP 2016), published by the ACM.

Success Factors in Managing Legacy System Evolution: A Case Study

Hennie Huijgens
Delft University of Technology and Goverdson, The Netherlands
h.k.m.huijgens@tudelft.nl

Arie van Deursen
Delft University of Technology
The Netherlands
Arie.vandeursen@tudelft.nl

Rini van Solingen
Delft University of Technology and Prowareness, The Netherlands
d.m.vansolingen@tudelft.nl

ABSTRACT
In this paper, we attempt to understand what contributes to a successful process for managing legacy system evolution. We provide an analysis of a number of key performance indicators such as cost, duration, and defects. By normalizing through function points, we furthermore compare to a larger benchmark. To do so we perform a mixed, retrospective case study on a series of nine software releases and eight single once-only releases, all performing on a single, legacy software system, in a West-European telecom company. We interviewed eleven stakeholders that were closely involved in the subject software releases. As a result, we list a number of observations from the quantitative and qualitative analysis. We found that a release process that performs above average on cost and duration satisfies stakeholders through fast response and direct value, even when the reliability and availability of the actual system is weak.

CCS Concepts
• General and reference → Cross-computing tools and techniques → Evaluation.

Keywords
Software Engineering Economics, Release-Based Software Engineering, Scrum, Cost Duration Matrix.

1. INTRODUCTION
Managing legacy systems, and especially linking the building of new software with evolution of legacy systems is a big challenge for many companies [1] [2]. For this study we analyze a series of nine software releases (the CECIL releases), performed in a West-European telecom company (in the remaining of this paper indicated as BELTEL), that is characterized by highly satisfied stakeholders. This study aims at analyzing software releases to only one system, that were conducted in different ways. The CECIL releases are typically built from quick wins; fast and small enhancements on a system by a single dedicated Scrum team. The releases all were performed on one Customer Relationship Management (CRM) system - named the DIVINE system, which is a five-year-old legacy system that is planned to be replaced because of ongoing reliability and availability problems.

While performing our case study, four things puzzled us. First, stakeholders were largely satisfied with the deliveries of the CECIL team. Second, the CECIL releases were assessed all to be 'best-in-class' in terms of cost, duration, and defects found, when benchmarked against other software deliveries in our research repository. Third, besides the CECIL releases another eight releases were performed on the DIVINE legacy system, outside of the scope of the CECIL team. And these were all assessed as not being 'best-in-class'. And finally, the DIVINE system itself was performing very badly in terms of reliability and availability.

The goal of this paper is to understand what contributes to a successful process for managing legacy system evolution. To reach this goal, we provide an analysis of a number of key performance indicators such as cost, duration, and defects. By normalizing through function points, we furthermore compare to a larger benchmark. Furthermore, to understand in depth what contributed to the success of the CECIL releases, we conduct in depth interviews with eleven people close involved with CECIL.

The remainder of this paper is organized in the following way: In Section 2 we outline the experimental setup for our case study. In Section 3 the research approach that we apply is described. Section 4 and 5 are about the results of our study. In Section 6 we discuss the results, compare them with state of the art and we discuss threats to validity. In Section 7 we discuss related work. Finally, Section 8 includes conclusions.

2. EXPERIMENTAL SETUP

2.1 Context
We analyze the CECIL releases and non-CECIL releases, performed over a period of one year on a single software system in BELTEL, a West-European telecom company. We perform both quantitative and qualitative analysis, the latter by performing open-ended, non-structured interviews with stakeholders on the backgrounds of the success of the releases. Regarding confidentiality of data, the names of companies, systems, releases, and people are made anonymous in this study. To improve the readability of this study, we provide definitions of four used acronyms:

CECIL releases: a series of nine software releases. CECIL releases are performed release-based, with a fixed team of six persons, with a steady heartbeat (Go Live every six weeks), and a Scrum approach. Within the CECIL releases only small enhancements are included; also in former years identified as CRM Quick Wins. These quick wins are primarily GUI-driven and meant to solve process issues in the DIVINE system that's mainly used by agents (front-office employees of BELTEL that have contact with customers through various channels (e.g. telephone, call centers, email, and chat). Driven by an attempt to speed up the software delivery process, in 2014 a decision was made to setup a fixed team

ICSSP'16, May 14-15, 2016, Austin, TX, USA
 ISBN 978-1-4503-4188-2/16/05…$15.00
DOI: http://dx.doi.org/10.1145/2904354.2904363

that was budgeted only once each year. This means that a budget was approved for capacity of the team for a whole year.

The CECIL release team consists of six people that all work fulltime for the team. From BELTEL itself these are a product owner and a business analyst, from BELTEL's main Indian supplier three software developers, and one tester from the main supplier that is responsible for user acceptance and regression testing. Besides that, an enterprise architect is involved in design activities on an ad hoc basis, and a release manager performs the integration once ready for release.

Non-CECIL releases: eight software releases on the DIVINE system that were performed once-only. Contrary to the CECIL releases these releases are characterized by a new team setup for every release, in advance governance and budget approval for each release, plan-driven approach.

The eight non-CECIL releases were performed as once-only releases, meaning that a team was setup preceding every single release and closed down once the release was finalized. Only few people within BELTEL were still to be found that joined in a non-CECIL release; of the interview participants mentioned in Table 4 only P1, P5, P8, P9, and P11 were involved in any way in these releases.

DIVINE system: The Customer Relationship Management (CRM) software system on which both the CECIL releases and non-CECIL releases are performed. A complicating factor in this study is the fact that the DIVINE system is a legacy system (planned to be replaced) that faces severe reliability and availability issues.

BELTEL: The West-European telecom company where CECIL releases and non-CECIL releases are performed. A repository with data of approximately 95 finalized software projects that is collected over time in BELTEL is used in this study as a reference for benchmarking purposes.

Within BELTEL a company standard allows for eight pre-planned production releases per year; each includes a number of projects and releases (from CECIL, DIVINE, or other teams), that jointly move on to user acceptance testing and integration into the production environment.

Following conventions in use at BELTEL, we use the term release with two different meanings. Release is used to indicate a specific software project that is performed in a release-based way. In this paper we use the term release for those cases. Besides that, the term release is used to indicate a combined set of projects and releases for integration into the production environment. In this paper we use the term *production release* if this is the case. Within the BELTEL practice, these *production releases* are deployed into the production environment in yearly eight subsequent production releases; this applies for both CECIL releases and non-CECIL releases too.

2.2 Research Questions

Based on this we define the following research questions:

RQ1: To what extent can a release-based iterative process be successfully used to manage the evolution of a legacy system?

RQ2: To what extent play known success factors a role in this success?

RQ3: What specific factors contributed to this successful way of managing legacy system evolution?

The case study that we perform is a mixed study: we perform both quantitative and qualitative research on the subject releases [3] [4]. The analysis falls apart in two parts. First, we quantitatively analyze the CECIL and non-CECIL release data that we collected on a series of releases over time and compare the outcomes with earlier research on best-in-class software releases [5]. Second, we conduct qualitative research by performing open-ended, non-structured interviews with members of the release-teams, and internal customers of BELTEL, that made use of the deliverables of the CECIL team and the non-CECIL teams.

2.3 Data Collection Procedure

For our research we make use of two types of data: data that was collected in the period before we started our study, as part of the operational measurement practice within BELTEL, and data we collect specifically for our research. The first consists of artifacts collected over time on these software releases, supplemented with data from two releases that finalized in the two previous years. Among others the following artifacts were available for our research:

- Quantitative data that was recorded in a measurement repository on both the CECIL releases and the non-CECIL releases, holding measurements such as size, cost, duration, number of defects, and planning data on cost and duration.
- Performance Dashboards and Project Close Reports on the finalized CECIL releases and non-CECIL releases.
- Logged data in the tool that was used for backlog management (Scrumwise[1]), including User Stories, Story Points, an activity log, attached documents, and comments on backlog items.
- Technical Design documents of the CECIL releases and non-CECIL releases, usually prepared by members of the CECIL team or members of the once-only non-CECIL releases teams

Besides collecting existing data within BELTEL we perform interviews with key stakeholders within BELTEL that are involved in the CECIL releases. The stakeholders are all involved in the operational practice of the CECIL releases; the list of stakeholders is setup in close cooperation with the business analyst and with the product owner. See Table 4 for an overview of interviewed stakeholders. All interviews are performed on a one-to-one basis between the first author of this paper [interviewer] and one specific stakeholder [interviewee], except for two interviews where two interview participants are combined in one interview at request of the participants. The interviews are based on the following questions, where applicable sub questions are asked to reveal backgrounds or to clarify misunderstandings or indistinctness:

1. Can you give some backgrounds on your role in the CECIL releases and non-CECIL releases?
2. What top-5 aspects did influence the releases in a positive way?
3. What top-5 issues need improvement?
4. In what way did the series of CECIL releases evolve over time (e.g. process changes, changes in way of working, team changes, changes in roles)?

[1] https://www.scrumwise.com

Table 1. Overview of the collected metrics for the CECIL releases.

	Go Live Date (mm-yyyy)	Release Size (FPs)	Story Points (Problem Tickets)	# User Stories (Problem Tickets)	Release Cost (Euros)	Release Duration (Months)	# Defects
CECIL 2014 R3	4-2014	111	na	na	56,345	3.12	11
CECIL 2014 R4	5-2014	37	na	na	45,769	2.76	5
CECIL 2014 R6	8-2014	64	31 (0)	16 (0)	30,367	4.60	2
CECIL 2014 R7	5-2014	80	64 (0)	17 (0)	73,134	5.49	8
CECIL 2014 R8	11-2014	40	37 (0)	6 (0)	58,154	7.56	7
CECIL 2015 R1	1-2015	46	62 (0)	12 (0)	84,464	6.11	19
CECIL 2015 R2	2-2015	109	87 (0)	18 (0)	62,768	7.43	7
CECIL 2015 R3	4-2015	128	73 (16)	23 (4)	62,964	8.81	16
CECIL 2015 R4	5-2015	63	44 (0)	8 (0)	110,000	6.11	10

5. In what way did the DIVINE system evolve over time (e.g. new functionality, enhancements, life cycle of the system)?
6. Are there any other important things to mention?

2.4 Quantitative analysis

In order to perform quantitative analysis on the collected data of CECIL- and non-CECIL releases, we calculate three performance indicators and compare the outcomes with those of our earlier published research paper on best-in-class software releases [5]. We calculate the following performance indicators:

1. *Cost per FP*; a weighted average of the summarized project cost in Euros divided by the summarized project size in FPs (weighting factor is project size).
2. *Duration per FP*; a weighted average of the summarized project duration in calendar days divided by the summarized project size in FPs (weighting factor is project size).
3. *Defects per FP*; a weighted average of the summarized number of defects found during each release divided by the summarized project size in FPs (weighting factor is project size).

In order to quantify the measure of success and failure we use a *Cost Duration Matrix*, a model built in earlier research [6] that compares the performance of finalized software projects in terms of cost, duration, and number of defects found during development. We compare the performance and characterizations of the CECIL releases with a series of 26 best-in-class software releases that were performed in another company and that we described in earlier research [5].

2.5 Qualitative Analysis

All interviews are recorded digitally and transcribed to text files. The text files are analyzed by following a number of steps. First, we read the transcripts, and make notes about first impressions. Subsequently, we label relevant pieces of the transcripts (coding) by labelling relevant words, phrases, sentences, or sections. We might decide that something is relevant for us because the interview participant explicitly states the importance, because it surprises us, or because it is repeated in several places. Our aim is to look for conceptualization and underlying patterns. Then we decide which codes are the most important, and create categories by combining codes to logical themes and drop less important ones. Finally, categories are labeled, and we decide which are the most relevant and how they are interconnected. We describe the connections between them. Finally, we discuss the impact and implications of our observations, based on the results of the quantitative and qualitative analysis. We use triangulating evidence from multiple sources to obtain our final findings.

3. QUANTITATIVE RESULTS

Between April 2014 and May 2015 nine CECIL releases are performed (see Table 1). Minimum release size was 37 FPs, maximum release size 128 FPs, medium size was 64 FPs. Cost of the releases varied from 30K Euro to 110K Euro, with a median of

Table 2. Overview of the collected metrics for the non-CECIL releases.

	Go Live Date (mm-yyyy)	Project Size (FPs)	Story Points	# User Stories	Project Cost (Euros)	Project Duration (Months)	# Defects
DIVINE Quick Wins 2012	11-2012	81	na	na	157,763	10.35	na
DIVINE Sales Orders	7-2013	110	na	na	358,883	11.60	18
DIVINE Detailed Rep.	10-2013	80	na	na	78,954	6.11	4
DIVINE Archival	12-2013	100	na	na	364,549	6.97	18
DIVINE Quick Wins 2013	12-2013	32	na	na	47,880	6.18	6
DIVINE Security	2-2014	61	na	na	181,483	6.93	5
DIVINE SO Tracking Tool	7-2014	22	na	na	69,761	9.20	7
DIVINE Stability Impr.	8-2014	8	na	na	22,930	8.31	na

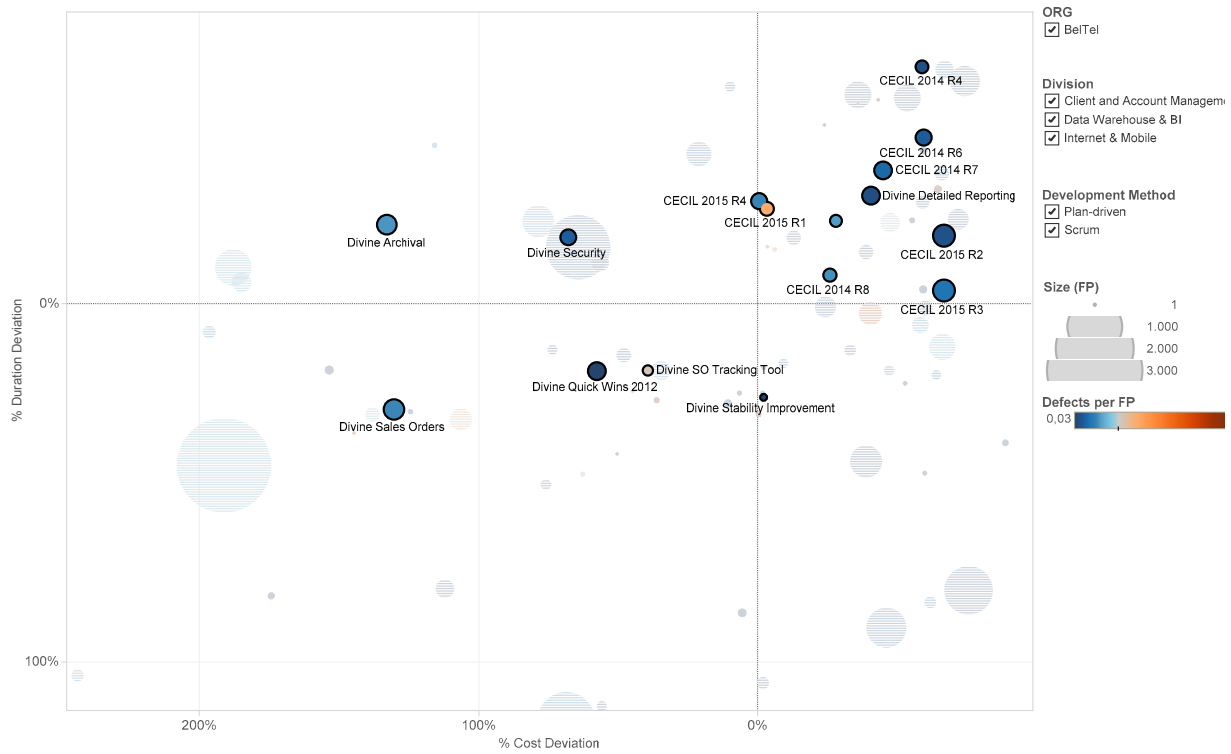

Figure. 1. Cost Duration Matrix of 95 finalized Beltel releases with the CECIL releases and non-CECIL releases mapped at it.

63K Euro. Release duration took 2.76 to 8.81 months, with a median of 6.11 months. Eight non-CECIL releases (see Table 2) were deployed. Minimum release size was 8 FPs; maximum size was 110 FPs. Cost varied from 23 to 365 K Euro. Duration took between 6.11 and 11.6 months.

Table 3 gives an overview of a comparison between quantitative data of collected within BELTEL on the finalized CECIL releases and 26 best-in-class software releases as analyzed in an earlier published research paper [5].

In order to benchmark the performance of CECIL releases and non-CECIL releases in terms of cost, duration and number of defects against our research repository holding 95 software releases that were performed within BELTEL, we use a model that we developed in earlier research; the *Cost Duration Matrix* [6]. The model can be used to compare a portfolio of releases to the benchmark, by means of a *Cost Duration Matrix*, as shown in Figure. 1 for the 17 releases in scope of this paper.

Each release is shown as a circle. The larger the circle, the larger the release is (in FPs). The 'redder' the release is, the more defects per FP it contains. The position of each release in the matrix represents the cost and duration deviation of the release relative to the benchmark, expressed as percentages. The horizontal and vertical 0%-lines represent zero deviation, i.e. releases that are exactly consistent with the benchmark. A release at (0%, 0%) would be one that behaves exactly in accordance with the benchmark; a release at (-100%, -100%) would cost nothing and be ready immediately; and a release at (+100%, +100%) would be twice as expensive and takes twice as long as expected from the benchmark. The 0%-lines divide the *Cost Duration Matrix* into four quadrants:

1. *Time over Cost* (top left); releases that score better than average for duration, yet worse than average for cost.
2. *Good Practice* (top right); projects that score better than average for both cost and duration.
3. *Cost over Time* (bottom right); projects that score better than average for cost, yet worse than average for duration.
4. *Bad Practice* (bottom left); projects that score worse than average for both cost and duration.

Overall, the quantitative analysis tells us that all CECIL releases fall in the *Good Practice* category when mapped on a *Cost Duration Matrix*, which means that all releases score better on *Cost per FP* and *Duration per FP* than the average of BELTEL projects. When compared with the 26 software releases that were performed within another company (see Table 3), we observe that CECIL releases are on average approximately two-and-a-half times bigger in size (FPs) than the best-in-class releases from earlier research [5]. Because of this and thanks to economies of scale the *Cost per FP* are approximately 10% to 20% lower. *Duration per FP* on the contrary is comparable to the average score of the best-in-class releases from [5]; average durations are longer, but the bigger average size in FPs compensates this.

Finally, the comparison shows that the number of *Defects per FP* of both CECIL releases and non-CECIL releases is higher than that of the best-in-class releases in [5], indicating a lower process quality. However, when benchmarked against our research repository

Defects per FP of all but one of both CECIL releases and non-CECIL releases is better than the average score.

Observation 1: The performance of CECIL releases is, in terms of Duration per FP equal to, and in terms of Cost per FP 10 to 20% better than that of the earlier described best-in-class releases.

When CECIL releases are compared with the performance of non-CECIL releases within BELTEL that are performed on the same DIVINE system, it shows that CECIL releases overall performed significantly better on both *Duration per FP* and *Cost per FP*. As possible explanations for this we assume that the additional startup time and cost needed for once-only releases, the learning effort and knowledge gap of non-CECIL once-only release teams, the extra time and cost for the proposal phase of a one-only release, and the overhead of once-only releases over a long-term, fixed, and experienced CECIL team play an important role. Besides that, we assume due to the type of requirements that a number of the non-CECIL releases are more nonfunctional than CECIL releases, leading to relatively less function points (non-functional requirements are not in scope of function point analysis).

Observation 2: The performance of CECIL releases is in terms of Cost per FP and Duration per FP, two to three times better than that of non-CECIL releases.

The quality of CECIL (0.13) and non-CECIL releases (0.14) in terms of *Defects per FP* is not as good as that of the best-in-class releases [5], yet still within boundaries when compared to the overall score of BELTEL as a whole (0.12). No indications are to be found in the quantitative figures that indicate too many defects during the development process.

Observation 3: Process quality in terms of Defects per FP of CECIL releases and non-CECIL releases is not as good as earlier described best-in-class releases from the full benchmark, yet on average when compared with BELTEL overall.

Besides the three performance indicators we also calculated two metrics that give us an impression of the availability and reliability of the DIVINE system, based on the tickets that were made on failures in the production environment in the first two quarters of 2015. Based on 32 tickets the *Mean Time To Repair* (MTTR) was 6:35, and the *Mean Time Between Failure* (MTBF) was 107:12, indicating that on average once every 4.5 day a failure occurs that lasts for on average 6.5 hours. The most worrying signal is that during repair the DIVINE system usually is not available for end-users and only limited sales can be performed by all shops of BELTEL.

Observation 4: The reliability and availability of the DIVINE system is worryingly bad and holds big risks for BELTEL's business continuity.

Concerning the success factors we identified in earlier research [5], we observe that three of them also apply to CECIL: a steady heartbeat, a fixed and experienced team, and release-based working on a single application. However, the factor of Scrum software delivery needs necessary differentiations.

The CECIL team certainly used a number of Scrum practices, such as a product owner, product backlog prioritization, and a product backlog management tool (Scrumwise) as a core instrument for its communication. And one may argue that although a product owner was distinguished, the availability and reliability of the DIVINE system was not included in this role, turning it into an information analyst with a Scrum label. Besides that, typical Scrum practices such as the role of the Scrum master and the daily standup meeting were not formalized within the team. Because of that we hesitate to label the CECIL releases as typically Scrum.

Table 3. Key Performance Indicator Comparison.

	CECIL Releases	non-CECIL releases	Best-in-Class Releases System A [5]	Best-in-Class Releases System B [5]
Number of Releases	9	8	13	13
Throughput (FPs)	678	494	415	349
Average Project Size (FPs)	75	62	32	27
Average Project Cost (Euros)	64,885	160,275	35,563	35,571
Average Project Duration (Months)	5.78	8.21	2.49	2.49
Average Number of Defects	9.44	9.67	na	1.23
Cost per FP (Euros)	861	2,595	1,114	1,325
Duration per FP (calendar days)	2.33	4.04	2.37	2.82
Defects per FP	0.13	0.14	na	0.05

Observation 5: Three success factors identified in earlier research apply to CECIL too: a steady heartbeat, a fixed and experienced team, and release-based working on a single application. However, the CECIL releases cannot be defined as typical Scrum.

For all non-CECIL releases we observe that only the factor release-based working on a single application applies. There was no fixed team, experience was not secured once releases were finalized, and a plan-driven (waterfall) delivery approach was followed.

4. RESULTS OF THE INTERVIEWS

In order to answer research question RQ2, 'what other factors can be found that influence release-based software delivery in a positive or negative way?' we performed nine interviews with eleven stakeholders. Table 4 gives an overview of the interview participants and their backgrounds. All interviewees are in one way or another involved in the CECIL releases, either as a user of the DIVINE system (business), as a member of the CECIL team, as a stakeholder from IT release management, or as enterprise architect responsible for the application landscape.

Overall the interviewees are more or less satisfied with the CECIL releases, as illustrated by a statement of interviewee P2 who revealed that requirements in CECIL are cherished with a typical nickname 'cecillekes' [Belgian colloquial for 'sweet Cecilia'].

In the following paragraphs we grouped aspects of observations from the interviews in nine categories.

Table 4. Overview of Interview Participants.

	Role	Business / IT
P1	Business analyst CECIL releases and (part of) non-CECIL releases	IT-department
P2 P3	Team leader Billing & Rating Support Billing & Rating support agent	Business Business
P4	Product owner CECIL releases	Business
P5	Release manager	IT-department
P6 P7	Consumer Care Mobile team leader Consumer Care Mobile team leader	Business Business
P8	Tester CECIL releases	IT-department
P9	Team leader Roadmap & Release Management	IT-department
P10	Shop Support & Channel Communication team leader	Business
P11	Enterprise architect – former team leader CRM team (a.o. DIVINE system and CECIL releases)	IT-department

Participants are depicted in one row when a combined interview took place (e.g. P2 and P3 were interviewed together).

1) Product owner is praised by many participants

The role of the product owner, and more specifically the way this role is fulfilled by the person in question, is in general highly appreciated; for stakeholders from business and IT alike. Many interview participants mention this as a first point when asked what aspects influenced the CECIL releases in a positive way. Examples are that the CECIL releases run very well (P2), and that the product Owner sets priorities within the backlog and determines the impact on the system (P1).

Observation 6: The role of the product owner and the specific personal fulfillment of that role is appreciated highly by stakeholders from both business and IT.

2) CECIL focuses on small but fast deliveries

Many stakeholders, especially those from business, mention as a success aspect that the CECIL approach focusses on quick wins; delivering high value for end-users (e.g. agents). Time-to-market is mentioned as a success factor (P5). We assume that the focus on delivering small enhancements in a fast and flexible way (P11, P1) is connected with this. Besides that, added value for the end-users (shops) is created by fast delivery with limited numbers of errors (P2).

"The idea of the CECIL items comes from the users. They see a bug in the system or a difficult process… I have to press this button twice or something… It is really based on ideas from the users. We discuss these ideas with [product owner] and see what can be put on the list." (P3)

Observation 7: CECIL is about quick wins: small, fast deliveries of requirements based on end-user problems.

A remark her can be made on the fact that the CECIL releases are applicable to one single application, the DIVINE system. One interviewee (P1) mentioned this as a success factor; referring to the fact that CECIL releases are independent from other teams. This finding corresponds with an observation from earlier research on two teams in a similar setting in another company [5].

3) Role of Scrum master is not formalized in practice

Contrary to that of the product owner, the role of the Scrum master is not formalized in the CECIL team. Despite the fact that at the start of the CECIL releases the team experimented with this role, it did not last in practice. Instead, the business analyst performs as an informal kind of coordinator in the team (P1).

"For me the most positive change was that [business analyst] joined the team. She replaced the former business analyst and she did a fantastic good job. She was just chasing [Indian supplier], she was keen on getting feedback and followed-up what was open." (P4)

Observation 8: The role of the Scrum master is not formalized in the team, yet no one seems to miss it.

4) Close cooperation within the CECIL team

Coordination within the CECIL team is an activity that is less formalized (e.g. there is no one with the formal role of Scrum master, coordinator, or project manager) yet the workflow seems to go smoothly with satisfied team members (P1). The transparent way of working together, and the open communication were mentioned as success factors (P7) The team members knew each other, lines were short, and the setting of the team was fixed and relatively small (P4). One time per week, or in the beginning even twice, a status meeting was organized (P1).

With regard to the fact that a part of the teamwork is done in India some remarks were made, although we did not get the impression that this was a big issue for the team itself. One Indian team member that works onsite acts as the main contact for the onsite team and as single point of contact for the offsite team members. Only small effects were observed here; handovers went quite smoothly (P4).

"An improvement was the replacement of [former developer from the Indian supplier] by [actual developer of the Indian supplier]. The former was most of the time in India and [actual developer of the Indian supplier] is most of his time here onsite. So that was more difficult. More conference calls, the sound was very bad, I didn't understand the language too good. Now we don't have fixed meetings anymore. We just walk by when needed and if it's convenient we setup a meeting." (P4)

Observation 9: Coordination is less formalized. It is a team activity; the CECIL team is a typical fixed and self-organizing team, although an onsite lead developer that coordinates offsite Indian team members helps a lot.

5) The Product Backlog management tool

A success aspect that is closely connected to the good cooperation within the team is the tool that is used for product backlog management: Scrumwise. Many interviewees mention it as very satisfying (P4, P5, P7). Stakeholders from business departments indicate that they use the tool for Kanban purposes in their own departments too. The tool supports good communication and bundles everything real-time together. People see right-away that someone is doing something (P4). It improves interaction between team-members and records all requirements and issues (5).

"Scrumwise is a good tool because it helps the team members to align activities. Everybody was available via the tool. How do we work? What are the agreements? It really had advantages for that purpose I think." (P7)

Observation 10: The product backlog management tool Scrumwise positively affected communication.

On documentation some small remarks were made that indicate that interviewees are satisfied with the level of technical design within CECIL, but that things need to be improved. However, this was experienced more as a general issue with regard to the Indian supplier (P1). Remarks were mainly regarding improvements on version control and technical design activities (P7).

6) Improvement: Budget and Estimating is fuzzy

The budgeting process with regard to the CECIL releases seems to be a bit fuzzy. Team members lack knowledge on what the budget is and how it is prepared. It is unclear how budgets are controlled and who is responsible for this (P4, P7).

With regard to estimation of new releases the same remark seems valid. The team does not use Story Points for estimating, but relies on estimates made by the developers of the Indian supplier: during the planning process they strongly advise on what requirements are in or out of a release (P4).

7) Improvement: Testing

Three parties are involved in testing. Build and integration testing is performed by offshore testers of the main Indian supplier that is responsible for development too. UAT and regression testing is performed by a tester within the CECIL team from an external supplier that is performing company-wide test activities for BELTEL. And finally testers from a Billing and Rating Support team perform a production test (P2).

Although all interviewees were unanimously satisfied with the CECIL process, some mention that testing can be improved, varying from smarter testing to follow-up of tickets recorded during the testing activities. Improvements were to be made in the somewhat informal, and laid-back approach on follow-up of test-tickets by the Indian supplier (P4).

Yet, although some interviewees indicate that testing needs improvement, some are quite satisfied about the quality as delivered by the CECIL releases.

"If you compare Cecil with other projects then Cecil scores much better in numbers of defects per test case. The pre-project period as delivering requirements and test cases goes very smoothly. Also the requirements are described very well." (P5)

Observation 11: Interviewees indicate that testing could be improved by smarter testing and better follow-up of test-tickets by developers of the Indian supplier.

8) Evolution of the process over time

All interviewed stakeholders are positive to very positive on the CECIL releases as they developed over time. Some interviewees even indicate that in case the DIVINE system is replaced in future, the process of enhancements as performed in the CECIL releases should be kept operational.

The team and its process is experienced as very stable and people know where to go with questions (P2), and it delivers small enhancements, but with high impact for business stakeholders (P4). A remark is made on whether the scope of the team should be enlarged to also larger enhancements too (including requirements that were in scope of once only projects that acted on the DIVINE system too (P7).

"As far as we are concerned Cecil should go on like it is…" (P3)

Observation 12: All interviewed stakeholders are satisfied about the CECIL way of working and want to have it continued in future.

9) Bad performance issues of the DIVINE system

As described earlier in Section 3 the DIVINE system suffers from severe problems with regard to availability and reliability. Many interviewees relate to this by mentioning that the DIVINE system is a legacy system, at the end of its lifecycle, mainly due to high costs for system upgrades by its original supplier and due to the ongoing availability and reliability issues.

"The serious stability problems of DIVINE are especially owing to database administration and integration with the backend. Those are the two things that need an architectural adjustment." (P11)

"The source of the performance problems is the fact that things are interwoven. If one application goes down most of the time fifteen others go down. Just replacing systems with the same functionality is only a good investment for the supplier and its partners. It's technical debt. But I call it also the blame game. We do a lot of system thinking instead of client thinking." (P7)

"The reason that DIVINE is at the end of its lifetime is that we wanted to do an upgrade, but the costs from the supplier of the system were very high. The system is also not that young anymore. Together with the cost the decision was taken to go for another system. In a way DIVINE is built as a CRM system. Yet BELTEL used it as a sales system with many customizations. It was not originally meant for that, and that's why there are many performance problems nowadays." (P1)

"DIVINE is not very stable, we have big performance issues. Loading problems, or error messages. I think that's the biggest reason to go to a new system. If you look at the high number of problems in the shops… They are talking to a customer and press a button and then they have to wait for two minutes… And all the Apache errors… The white screens when you have to completely log off and start again…" (P3)

Besides performance issues also the fact that the DIVINE system functionally evolved in a difficult to maintain solution for the business users is mentioned as a problem for the future.

"Functionally DIVINE works as it works… we made things wrong ourselves. We rebuilt things and Cecil could stick some plasters on some wounds. But there is no 'wow' to make out of it anymore…" (P4)

A decision is taken to replace the DIVINE system by a new product that will be a package off-the-shelf solution with minimal customization and minimal integration with backend systems.

"In former times DIVINE went down every day. Things improved. But do we have a proper CRM? No! That's why the system is going to be replaced in the coming six months. We are now looking at a new package solution with as little connections to Provisioning and Billing systems to make the dependencies as small as possible." (P11)

Yet, it is interesting to observe that stakeholders tend to judge the CECIL releases and the DIVINE system as different things that are not interrelated.

"Cecil stands loose from the system that does not work. [Interviewer]: But why are there no performance improvement issues on the Cecil backlog? [Interviewee]: Well they tried all kind

of things to improve the performance. I don't think that's going to be a quick win. A specialist came over from the USA to see how he could solve things. If Cecil was planned to solve these problems, then the management would already put performance things on the backlog." (P2)

"This morning we had a meeting were the operations manager opened his heart on an issue last month, when DIVINE was down for one day due to firewall issues, and because of that BELTEL did not make any money for a whole day. It is interesting why these non-functional aspects are not incorporated into Cecil. Apparently nobody thinks about this" (P5)

"It is a classic problem within BELTEL that Operations asks to be involved in a project… But effectively they only react per email on questions. We look at DevOps, but we are not even agile yet. What do you expect?" (P11)

Maybe the most striking finding of our analysis, is the fact that it is possible to have stakeholders that are all quite satisfied with the process of releasing a steady stream of enhancements over time, yet on the other hand they have to struggle with a software system that is lacking in reliability and availability.

Observation 13: A release process that performs better than average on cost and duration, and on average on defects, can satisfy stakeholders in managing changes on a badly performing software system.

5. DISCUSSION

5.1 Threats of Validity

With regard to construct validity, the degree to which a test measures what it claims to be measuring a remark is in place on Function Point Analysis. Functional documentation is used to count FPs, holding the consequence that low quality documentation could have led to low quality FPA. To mitigate this risk, we thoroughly reviewed all sets on completeness and correctness, we made use of two certified FPA specialists, and assured that all involved FPA specialists are trained and experienced. To prevent from using low quality release data, we had all data reviewed by the product owner and the business analyst and the financial controller of BELTEL.

By normalizing all project data with the functional size in FPs we warranted internal validity, the extent to which a causal conclusion is based on our study. By doing so we could more objectively compare performances of all releases in order to minimize systematic error. The effect of outliers is limited and the risk on bias is mitigated responsibly based on the diversity of projects and business domains within BELTEL, the number of software projects, and the fact that we measured and analyzed software project portfolios as a whole in an empirical way.

With regard to external validity, whether the study results can be generalized to settings outside the study, we argue that due to the limited scope of our study, one specific series of releases in one company, it is too early to generalize the outcomes. Since we looked at seventeen releases performed in one company; the outcomes cannot be generalized to other environments without precautions. A promising factor here is however, that we compare the quantitative results of our study with results from existing research that we performed on similar software releases in another company, leading to a general expectation that the outcomes of our study might generalize to similar release approaches and companies too and that factors such as a steady heartbeat, release-based way of working mapped on a single application, and a fixed and experienced team are common success generators for software engineering.

5.2 Scrum as a Distinguishing Factor

Our research did not indicate that Scrum in itself is a distinguishing factor for the success of the software releases, but that some specific elements of Scrum, namely, the role of the product owner, a product backlog management tool, and short iterations based on prioritized requirements that deliver high value to end-users, are key elements that lead to release processes that outperform on cost and duration. The role of the Scrum master, daily standup-meetings, planning and estimations in Story Points, and the concept of Sprint Reviews were not adopted. As such, this setting should not be qualified as a Scrum setting, but as a local agile implementation using some Scrum practices. Furthermore, the fact that the CECIL team spends time on upfront tasks such as writing a design document deviates from many agile approaches.

5.3 Impact / Implications

With regard to our first research question we conclude that the CECIL releases corresponds to four of the five success factors mentioned in earlier research [5], a steady heartbeat (taking into account that the duration of releases varied, but the release dates were preset upfront), release-based working on a single application, and a fixed and experienced team. The success factor Scrum did not fully correspond with the subject CECIL releases, but a number of Scrum practices were in place. The non-CECIL releases only correspond with one factor, namely release-based working on a single application.

Analysis of the observations from the interviews, based on a grouping of observations and connections in coherent categories, reveals three categories with regard to the CECIL releases: the CECIL team, the CECIL release performance, and the DIVINE system. Our study indicates that these categories apparently can live together without very close connections. The CECIL team itself performs very well, stakeholders are quite satisfied, while the subject DIVINE system performs poorly in terms of both reliability- and availability.

For future use the model that we used to benchmark the performance of releases against our research repository, the *Cost Duration Matrix*, should be expanded with metrics on the performance of software systems after deployment in a production environment. In this way concepts such as *Good Practice* and *Bad Practice* will reflect the performance of software releases in a more realistic way.

6. RELATED WORK

Challenges with legacy systems as described in our case study are examined in many related work. Boehm [1], for example, mentions legacy evolution as one of the major future challenges for systems and software dependability processes. Van Deursen et al. [2] see "to try to bridge the gap between research aimed at building new software and research aimed at maintaining or renovating old software" as a large challenge.

A common idea of many research performed in the former millennium is that success and failure are interconnected with process-based activities [7]. Reel [8] mentions five critical success factors in software projects, such as start on the right foot, maintain momentum, track progress, make smart decisions, and institutionalize post-mortem analyses. Dybå [9] [10] analyzed success factors for software process improvement, such as business orientation, leadership involvement, employee participation,

concern for measurement, and exploitation of existing knowledge. Niazi et al. [11] identifies a large number of success factors for software process improvement implementation in existing literature. Rainer and Hall [12] surveyed practitioners and found factors such as reviews, standards and procedures, training and mentoring, and experienced staff that practitioners generally considered had a major impact on successfully implementing SPI. Besides that, they found factors such as internal leadership, inspections, executive support and internal process ownership that the more mature companies considered had a major impact on successfully implementing SPI. Stelzer and Mellis [13] mention ten success factors of organizational change in software process improvement, a.o. management commitment and support, staff involvement, and providing enhanced understanding.

More recent work emphasizes the success and fail factors of shorter iterations due to an agile way of working. Chow and Cao [14] surveyed agile professionals on success in agile software projects, and came up with factors such as delivery strategy, agile software engineering techniques, and team capability. Misra et al. [15] found factors such as customer satisfaction, customer collaboration, customer commitment, decision time, corporate culture, control, personal characteristics, societal culture, and training and learning. Sutherland et al. [16] assessed hyper productivity in Scrum and mentions success factors such as team formation, Scrum meetings, sprints, product specification, testing, configuration management, pair programming, and measuring progress as typical for success.

Meyer [17] identifies a number of contributions of the agile approach: refactoring, short daily meetings that support good team communication, identifying and removing impediments, and identification of sources of waste. As “brilliant agile principles” he mentions short iterations, continuous integration, the close window rule (no functionality can be added during an iteration), time boxing, the role of the product owner, an emphasis on delivering working software, the notion of velocity, and associating a test with every piece of functionality [17].

Colares et al. [18] present an approach to the software release planning problem, based on a mathematical formulation that is based on the idea to maximize stakeholder satisfaction and to minimize risks. Lehman and Ramil [19] define eight laws on software evolution: continuing change, increasing complexity, self-regulation, conservation of organizational stability, conservation of familiarity, continuing growth, declining quality, and feedback system.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we address the problem of different ways of working for evolving legacy software, which by definition resist change. The contributions of this paper are threefold. First, it gives a description of a case in which a release-based, iterative process on a legacy system worked well, satisfying key stakeholders despite the poor quality of the system itself. Second, it confirmed three success factors as identified in our earlier research as contributors to this success [5]:

1. A steady heartbeat;
2. A fixed and experienced team
3. A release-based way or working, mapped on a single system.

Third, we identified four additional success factors:

4. The role of the product owner and the personal interpretation of that role;
5. A focus on quick wins and small, fast deliveries of requirements based on end-user problems;
6. The fact that the role of the Scrum master is not formalized, leading to a self-organizing team with an onsite lead developer that coordinates offsite Indian team members;
7. A specific product backlog management tool that positively influences communication.

The research as presented opens prospects for future research. With regard to our observation that the subject CECIL team applies less formalized aspects of Scrum in its process, we conclude it is important to examine whether the findings from this study are applicable to teams that work according to more formalized settings of Scrum too.

Finally, in interviews we heard stories about poor performance of the DIVINE system. Since we considered this to be out-of-scope for this study we checked this finding only quantitatively. For future studies we will include qualitative analysis of the underlying legacy system in our study too.

ACKNOWLEDGMENTS

We thank BELTEL for its generosity to allow us to use company data for our research, and the members of the CECIL team, and other interviewed stakeholders for their willingness and openness to share their experiences with us.

REFERENCES

[1] B. Boehm, "Some future trends and implications for systems and software engineering processes," *Systems Engineering,* vol. 9, no. 1, pp. 1-19, 2006.

[2] A. v. Deursen, P. Klint and C. Verhoef, Research issues in the renovation of legacy systems, Springer Berlin Heidelberg, 1999.

[3] P. Runeson, M. Host, A. Rainer and B. Regnell, Case Study Research in Software Engineering; Guidelines and Examples, Hoboken, New Jersey. USA: John Wily & Sons, 2012.

[4] R. Yin, Case Study Research - Design and Methods, Los Angelos, USA: Sage Publications, 2008.

[5] H. Huijgens and R. v. Solingen, "Measuring Best-in-Class Software Releases," *IWSM-MENSURA 2013 Joint Conference of the 23rd International Workshop on Software Measurement and the 2013 Eighth International Conference on Software Process and Product Measurement,* no. IEEE, pp. 137-146, 2013.

[6] H. Huijgens, R. v. Solingen and A. v. Deursen, "How to build a good practice software project portfolio?," in *ACM Companion Proceedings of the 36th International Conference on Software Engineering (ICSE-SEIP)*, 2014.

[7] T. Hall, A. Rainer and N. Baddoo, "Implementing Software Process Improvement: An Empirical Study," *Software Process Improvement and Practice,* vol. 7, pp. 3-15, 2002.

[8] J. Reel, "Critical Success Factors in Software Projects," *IEEE Software,* Vols. May-June, pp. 18-23, 1999.

[9] T. Dybå, "An Empirical Investigation of the Key Factors for Success in Software Process Improvement," *IEEE*

Transactions on Software Engineering, vol. 31, no. 5, pp. 410-424, 2005.

[10] T. Dybå, "Factors of Software Process Improvement Success in Small and Large Organizations: an Emperical Study in the Scandinavian Context," in *ESEC/FSE*, Helsinki, Finland, 2003.

[11] M. Niazi, D. Wilson and D. Zowgh, "Critical Success Factors for Software Process Improvement Implementation: An Empirical Study," *Software Process Improvement and Practice,* vol. 11, pp. 193-211, 2006.

[12] A. Rainer and T. Hall, "Key success factors for implementing software process improvement: a maturity-based analysis," *Journal of Systems and Software,* vol. 62, no. 2, pp. 71-84, 2002.

[13] D. Stelzer and W. Mellis, "Success Factors of Organizational Change in Software Process Improvement," *Software Process Improvement and Practice,* vol. 4, pp. 227-250, 1998.

[14] T. Chow and D.-B. Cao, "A survey study of critical success factors in agile software projects," *The Journal of Systems and Software,* vol. 81, pp. 961-971, 2008.

[15] S. C. Misra, V. Kumar and U. Kumar, "Identifying some important success factors in adopting agile software development practices," *The Journal of Systems and Software,* vol. 82, pp. 1869-1890, 2009.

[16] J. Sutherland, A. Viktorov, J. Blount and N. Puntikov, "Distributed Scrum: Agile Project Management with Outsourced Development Teams," in *40th International Conference on System Sciences*, Hawaii, 2007.

[17] B. Meyer, Agile!: The Good, the Hype and the Ugly, Springer Science & Business Media, 2014.

[18] F. Colares, J. Souza, R. Carmo, C. Pádua and G. Mateus, "A new approach to the software release planning," in *IEEE XXIII Brazilian Symposium on Software Engineering (SBES)*, 2009.

[19] M. M. Lehman and J. F. Ramil, "Rules and tools for software evolution planning and management," *Springer, Annals of software engineering,* vol. 11, no. 1, pp. 15-44, 2001.

TUD-SERG-2016-009
ISSN 1872-5392