Does Software Modernization Deliver What It Aimed for? A Post Modernization Analysis of Five Software Modernization Case Studies

Ravi Khadka*, Prajan Shrestha†, Bart Klein*, Amir Saeidi*, Jurriaan Hage*,
Slinger Jansen*, Edwin van Dis‡, Magiel Bruntink†
*Utrecht University, The Netherlands
{r.khadka, bart.klein, a.m.saeidi, slinger.jansen, j.hage}@uu.nl
†University of Amsterdam, The Netherlands
p.shrestha@students.uva.nl, m.bruntink@uva.nl
‡CGI, The Netherlands
edwin.van.dis@cgi.com

***Abstract*—Software modernization has been extensively researched, primarily focusing on observing the associated phenomena, and providing technical solutions to facilitate the modernization process. Software modernization is claimed to be successful when the modernization is completed using those technical solutions. Very limited research, if any, is reported with an aim at documenting the post-modernization impacts, i.e., whether any of the pre-modernization business goals are in fact achieved after modernization. In this research, we attempt to address this relative absence of empirical study through five retrospective software modernization case studies. We use an explanatory case study approach to document the pre-modernization business goals, and to decide whether those goals have been achieved. The intended benefits for each of the five cases we considered were all (partially) met, and in most cases fully. Moreover, many cases exhibited a number of unintended benefits, and some reported detrimental effects of modernization.**

I. Introduction

In the software engineering, it is widely accepted that real world software must be continually adapted or enhanced to remain operational [1]. The need for constant adaptation or enhancement within an operational software system is triggered by various factors such as adapting to new business requirements, changes in legislation, advancement in technology [2]. Lehman's laws of software evolution suggest that operational software systems must often reflect these changes, otherwise they become progressively less useful to the stakeholder [1]. Hence, evolving these operational software systems by constantly adapting to changes is critical and requires significant resources [3], [4]. Failure to take remedial changes gradually makes them costly to operate and maintain, thereby turning them into legacy software systems– systems that significantly resist modification and are less maintainable [5].

In terms of the software life cycle, Comella et al. [6] categorize software evolution into three activities: maintenance, modernization, and replacement. Despite the fact that some researchers and practitioners use software evolution and software maintenance interchangeably [4], this research distinguishes these two terms and instead adopts the categorization (i.e., maintenance, modernization, and replacement) proposed by Comella et al. [6].

Software modernization has been extensively researched in academia, primarily to increase maintainability, increase flexibility and reduce costs [4]. Hence, a plethora of software modernization methods exist. The majority of these aim to address technical aspects of modernization, i.e., providing technical solutions to perform or to facilitate the software modernization process [7]. Furthermore, these technical solutions of software modernization are labeled as "successful" once the modernization process is proven to be technically feasible. As per our knowledge, very limited, if any, (empirical) assessments of the impact of software modernization in terms of business goals exist. Nasr, Gross & van Deursen [8] indicate the need for a distinction between the technically feasible outcome and the impact of software modernization. Lack of empirical evidences of impact software modernization can contribute to different expectations and wrong estimations of resources. This knowledge gap can potentially lead to delays or even failures of software modernization projects. However, measuring the impacts of software modernization is not trivial.

In this paper, we present five retrospective case studies of software modernization with the aim of documenting the impacts of modernization. We adopt an explanatory case study research method (seeking an explanation of a situation or problem for pre- and post-event studies [9]) to explore the pre- and post- modernization situation. The contribution of this research is two-fold: first, it documents the expected benefits and the impacts of software modernization by analyzing pre- and post- modernization situations, and second, it compares the post modernization impacts with the benefits of software modernization as claimed by academic research.

Section II of this paper reviews related work. Section III presents the case study setup and research method used for this research. Section IV provides the five case study reports. Section V presents the findings of the cases. Section VI discusses the threats to validity of the research. Finally, Section VII provides concluding remarks and relevant future research directions.

II. Related Work

A. *Software Evolution and Software Modernization*

Software evolution and software maintenance are mature research domains in software engineering. Despite, several researchers and practitioners use software evolution as a preferable substitute for software maintenance [4], this paper distinguishes these two terms. Godfrey & German [10] provide a distinction between software evolution and software maintenance; the former is used to describe the phenomena associated with modifying existing software systems, whereas the latter describes the activities to make the existing software systems easier to manage and change. Adhering to this distinction between software evolution and software maintenance, we adopt the concept of software evolution discussed by Comella et al. [6] in terms of the software life cycle. They categorize software evolution activities into three: maintenance, modernization, and replacement. When a software system is deployed, maintenance activities are used to keep it operational. But, as the software system becomes increasingly outdated, maintenance becomes too challenging and costly, thereby requiring a modernization effort to do extensive changes rather than maintenance. Finally, when the old system can no longer be evolved, it is then replaced. Hence, software modernization is the process of evolving existing software systems by replacing, re-developing, reusing, or migrating the software components and platforms, when traditional maintenance practices can no longer achieve the desired results [11].

Research within software evolution and software maintenance has been primarily focused on building an understanding of software evolution and maintenance by (empirically) observing the phenomena [10]. Such observations include studies of large scale industrial software systems (e.g., Belady & Lehman et al. [1], Gall et al. [12]), and open source systems (e.g., Godfrey & Tu [13], Koch [14]). Furthermore, empirical methods such as surveys (e.g., Kemerer & Slaughter [15], Kagdi et al. [16]) have been used to understand software evolution. A summary of empirical studies performed to understand open source software evolution is reported by Fernandez-Ramil et al. [17]. Various methods and techniques have been used to understand software evolution such as a change-based approaches (e.g., Robbes & Lanza [18]), software visualization techniques (e.g., Lanza [19]), program transformation (e.g., Baxter [20]), mining software repositories (e.g., Kagdi et al. [16]). Similarly, numerous (empirical) studies have been reported to understand the software maintenance phenomenon (e.g., Singer [21], Bianche et al. [22]). A plethora of software modernization approaches have been published, including literature reviews. For instance, Comella et al. [6] reported black-box modernization approaches; Razavian & Lago [23], Almonaies et al. [24] published reviews on migrating legacy systems to SOA; and recently Jamshidi et al. [25] reported legacy to cloud modernization approaches.

Despite this large number of (empirical) research, efforts have been focused on observing the phenomena [10] [2], and addressing technical aspects to facilitate software evolution, maintenance and modernization [7].

B. *Benefits of Software Modernization*

For the past three decades, the software evolution community has proposed several methods to modernize legacy software systems. These modernization methods aim at helping organizations achieve various benefits such as reduced costs, increased flexibility, and improved maintainability. To get a systematic overview of such benefits as suggested in academia, we conducted a literature review using a backward snowballing approach [26], i.e., using the reference list of a paper to identify new papers to include. Table I depicts a non-exhaustive list of benefits, referred to as "claimed benefits" hereinafter, as a result of backward snowballing. We have observed that there is a clear absence of empirical research that measures these benefits, although limited research has been validated with industrial case studies to assess the applicability.

TABLE I
Claimed benefits identified in the literature

Claimed Benefits	**Research Paper**
Cost reduction	[8], [24], [7], [11], [27], [28], [29], [30], [31], [32], [33], [34], [35], [4], [36], [37], [38]
Increased reusability	[8], [24], [29], [30], [34], [35], [37], [39]
Increased agility	[8], [24], [23], [29], [30], [40], [35], [39]
Increased flexibility	[8], [24], [23], [11], [28], [32], [37]
Improved performance	[29], [40], [41], [33], [34], [35], [37]
Increased maintainability	[28], [30], [31], [42], [36], [38]
To remain competitive	[25], [30], [43], [33], [34], [37]
Increased availability	[11], [25], [30], [31], [33], [36]
Faster-time-to-market	[23], [11], [40], [4], [36]
Increased interoperability	[7], [25], [30], [31], [44]

To summarize, there has been a very limited, if any, research assessing the post-modernization situation and there is a need for empirical research in collaboration with industry to assess whether any of the claimed benefits are met [8]. In this paper we address the two key gaps in current understanding: the lack of empirical case study research documenting the claimed benefits as established in the pre-modernization phase, and to which extent those benefits were in fact met after modernization.

III. Case Study Design

We report on five retrospective case studies of software modernization. We chose retrospective cases for several reasons. First, the objective of this research explicitly requires successful software modernization cases that have been performed in the past. Second, the retrospective nature makes it possible to get an in-depth understanding of a contemporary phenomenon where the investigator has little control over events, thereby reducing research bias [45]. We have adopted an explanatory case study research method, primarily seeking the rationale for initiating software modernization and documenting the impacts of software modernization. Despite the fact that case studies are originally used primarily for exploratory purposes [46], Runeson & Host [9] argue that explanatory case studies are more suitable for investigating the pre and post-event situations. Our research aims at documenting pre- and post-modernization situations, and thereby fits well with the latter.

Data collection in this case study is performed by: (i) consulting documentation to identify the need for and goals of the software modernization, and (ii) semi-structured interviews to understand the rationale and impacts of software modernization. To identify the objectives of the modernization, we started with consulting project documents including, but not limited to, project initiation documents (PIDs), business cases for modernization, modernization strategy documents, project management reports, and intermediate milestone deliverables, whenever available. As for semi-structured interviews, we conducted nine interviews. The interviews were informal and, whilst grouped around three themes, were designed to allow the conversation to follow the respondents' interests. The interviewees typically included a project manager and/or a technical manager. In one of the cases, we also interviewed an IT director and a finance manager. We decided to involve people with different roles to obtain a varied outlook on the situation. For instance, an IT Director potentially provides a better rationale/business case for software modernization while a technical manager can elaborate on the technical issues and impacts of modernization. Prior to the interviews, each interviewee was introduced to an interview protocol, a document detailing the objectives of the interview with relevant questions grouped into themes, and a glossary of the technical terms to attain a common understanding. The interview protocol included a brief introduction to the research, questions regarding personal background of the interviewee (name, current role and responsibility, expertise, and experience), and interview questions categorized into three themes: pre-modernization situation, modernization process, and post-modernization situation.

TABLE II
DETAILS OF THE INTERVIEWEES

SNo	Role	Experience	Company
P1	Technical Lead	34	Infra Co.
P2	Developer	29	
P3	Finance Manager	28	
P4	ICT Business Coordinator	13	Aviation Co.
P5	Business System Analyst	34	
P6	IT Director	30	
P7	Project Manager	10	Public Service
P8	Migration Lead	24	Gov. Office
P9	Project Manager	20	Finance Co.

All the interviews were conducted in English and lasted between 60-120 minutes, except one that was conducted in Dutch and later translated to English. All the interviews were conducted either in-person or via skype (for the international case studies). These interviews were recorded and then transcribed. In case more information or clarifications were needed, the interviewee and/or relevant sources identified by the interviewee were consulted in-person or via email. Table II depicts the anonymized details of the interviewees with role, years of IT experiences, and the domain of the company. Nvivo 10[1] is used as an instrumentation tool to facilitate the interview analysis process. After the individual case studies were analyzed, we used cross-case analysis (CCA) [45], [47], a data analysis method that analyzes multiples cases studies seeking for empirical evidence on a specific fact, synthesizing data, drawing inferences, and providing recommendations [45], [48]. In this research, CCA is used to identify similarities and differences among the case studies to develop concrete findings based on them.

IV. Case Studies

We analyzed five software modernization cases within Europe that were completed at least two years ago. Four of them are based in the Netherlands and one in Portugal. The case companies come from different domains: two are from the public sector (Government organizations), two are industrial companies, and one is a financial company. We start with identifying the pre-modernization business goals, referred to as "expected benefits" hereinafter and then document the impact after modernization, referred to as "observed benefits". We do not focus on the modernization process itself. For each case study, we provide a brief summary of the case company and the pre-modernization scenario, list the expected benefits, and describe the impacts after modernization. The documented benefits are supported by relevant data i.e., interview quotes and relevant texts from the documentation. To anonymize the products, [**legacy system**] and [**modernized system**] are used in the quotes and some textual corrections are made within [], whenever necessary to increase understandability.

To provide an impression of the size of the legacy systems prior modernization, Table III depicts some of the available details. For the "Public Service" case, details of the system were not available.

TABLE III
DETAILS OF THE LEGACY SYSTEMS

Company	Language		
	Name	Size (LoC)	# Module
Infra Co.	Progress	over 50K	Not Available
Aviation Co.	Progress	51370	1803
Public Service	COTS	Not Available	Not Available
Gov. Office	COBOL	over 1 million	2500
Finance Co.	COBOL	9.289 millions	3548

A. *Case I: The Infra Co. Case*

The Infra Co. is a company based in the Netherlands that develops innovative solutions for the consumer market, in particular, installation, construction and special products. The company had a COBOL-based legacy information system (IS) that was migrated to Progress[2] in 1990. The Progress-based IS was a highly performant character-based application with sales order, purchase order, warehouse management, financial management and marketing management modules. The IS was a client-server based application running on SUSE Linux on

[1] www.qsrinternational.com/products_nvivo.aspx
[2] Currently known as ABL–www.progress.com/openedge/features/abl

HP hardware. In 2007, Infra Co. started a project to modernize the existing Progress-based IS from console-based to GUI and to re-host the IS in VMware-based virtualized servers. The modernization project lasted for around 12 months. This case description is based on the latter modernization project.

1) Enhance Usability: The primary objective was to modernize the existing character-based user interface system. Furthermore, incremental development with different departments led to 3 different user interfaces for the same backend. Aiming to improve the user interface, the case company initiated the modernization process. P1 expressed this as *"In our case the modernization was mainly the modernization of user interface because we are thinking for 90% we could re-use the code which was behind the user interface."* P2 emphasized the need of modernization as *"We encountered many problems so we decided to modernize our [applications] which were reliable character user interface. And [we] modernize it into [**modernized system**]."*

With the modernization, the company transformed their character-based application to [**modernized system**] based GUI. P2 mentioned *"The goal was to get one interface that succeeded. Flexibility is good because it is easy and fast to change anything within [**modernized system**]. So that's easy. [..] I think we have reached what we wanted, I think. I think the users are happy with the programs."* P1 expressed the impact of enhanced usability. For instance, he stated *"But users got used to it very quickly [..] It's nice to work with [...] I think much more user friendly [...] key of F1 was same everywhere, the buttons were with same functions and they got used to it very quickly."* P2 supported P1 as *"After working few weeks with new system, it gets easy [..] started getting used to the system very fast."*

2) Product Consolidation: The Infra. Co. has a subsidiary in Belgium that has its own application portfolio. Historically, the applications were the same but over time incremental developments in the Netherlands and Belgium resulted in diverse applications accessing the same data. The company used this modernization to consolidate these applications into one. P1 expressed this as *"We started with this character [based] system and then [..] we started to program with the [legacy system] in windows environment and again 5 years later we started with [**modernized system**] but we never converted old software with [to] the new one. So we ended up with 3 systems next to each other and they all have different functions while they use the same database and you can say that programs used for finances were character interface programs, programs used of sales department were [**legacy system**] tools [..] and the programs for purchase department were created in [different platform]."* P2 indicated that it was important to consolidate those applications as *"of course it was different environment, [we want to] bring it back to one environment. [**legacy system**] has its own databases, [different platform] has own, we have our own database. So that is also an important issue [..] and for maintenance, updates; it is easier to have one rather than 3 environments."*

After modernization, the company has now consolidated the application into one with new interfaces, a separate business logic layer and a database layer. P1 mentioned that with the product consolidation, the development team has benefited *"Well it [Legacy application] was bit messy for users so [we] cleaned up and [now we have] one nice system in which we could develop more programs."* He also indicated that testing of application has become significantly easier as *"So testing is far more easy."*

3) Increase Maintainability: Maintenance of the legacy system was difficult as mentioned by P1 as *"the [**legacy system**] software, maintaining was very very hard for us, for development point of view."* The difficulty to maintain the legacy systems was due to the fact that individual sister companies were running their own silo systems. P1 expressed that as *"we had 3 companies with 3 different environments and also with also different Progress version, it was very hard to maintain."* P2 further explained *"So we had 3 interfaces and some people had 3 buttons in[on] screen and they have to open all 3 to use all the programs they have. That was not easy for maintenance."*

The new system turned out to be highly maintainable as compared to the legacy application. P2 mentioned this as *"It is easier to maintain programs, biggest advantage I mean."* The impact of modernization on increased maintainability was expressed by P1 as *"so we had to get all things in one table as this was not easy to maintain 2 tables for the same thing. That was difficult in the beginning."* With the new system, modification of the programs for end users was simple to achieve. P1 expressed this as *"About maintainability, [it] is quite easy. We did a lot of work on, it's a one second [one second of] work, if he needs [an] extra right, it's very easy to do."*

4) Unintended Benefits and Detrimental Effects: After modernization, transparency in the organization has increased. End users are more clear on what their daily job is, and collaboration has been on the rise. P1 mentioned *"For users, it doesn't become more flexible but more clear."* and P3 as *"departments are using the same system to collaborate and it is much better than before. Yes, for more transparency."* Furthermore, the company has achieved a maintenance cost reduction by reducing the number of programmers from 5 to 2 and lower maintenance activities. P1 expressed this as *"So there were 5 persons who could develop on that system and now we are only 2 [...] So the cost reduction is mainly in less time we spent on maintenance of the system."*

When it comes to detrimental effects, the company did observe some user resistance in the initial stages. The users were used to the character-based interface. P2 indicated this behavior as *"Most users were not really happy when going to the Windows [**modernized system**] surroundings [environment]. Because they were so used to those [character-based] screens."* P1 raised concerns over performance and mentioned that the new system was not as performant as the legacy system and said *"About performance, well it's not really bad but its not really fast but fast enough [..] Because it's a character interface the performance is good and per definition faster* *than windows. And that was the biggest advantage."*

B. Case II: The Aviation Co. Case

The Aviation Co., headquartered in the Netherlands, excels in the aerospace market with over 50 years of experience in selling aircraft parts to customers in more than 100 countries. The company had a "Quotation Management" application built around 2001 that was running on the three continents where the company has branches. The application used a Progress database and was coupled with three other information systems within the company. The modernization project lasted for around 11 months.

1) Increase Flexibility: The primary goal of this modernization was to enhance flexibility by decomposing the monolithic legacy system, in order to increase the possibilities of reuse of system modules. P5 expressed this as *"So breaking up into reusable modules, and splitting layers like user interface, business logic, database access. [..] and in separate re-usable modules as well, so we can interchange and allowing business logic which is proprietary to the business systems to be accessed by outside parties like customers or suppliers."* P5 sees this as an opportunity to expand their business by making the existing landscape more flexible. He says *"Before, we were combining business logic, screens and database layers to the same setup code, [...] or when you develop a new application we can reuse the business logic layer in the form of a black box and also consuming from another application so we made our software also more flexible for future developments."*

After successful modernization, the company not only decomposed their legacy landscape but also started offering their modules as services (i.e., Software-as-a-Service). P4 mentioned that the new system is able to provide services as *"[...]one case where we have to use modernization also[is] to package the application to provide it as a service to one of our customers. Before we were the only user of it. And we had developed an in–house inventory tool but now we were selling it as a service, Software-as-a-Service also to one of our customer."*

2) Increase Maintainability: One of the key business goals was to increase maintainability of the legacy systems. The legacy systems were running independently in three geographical locations (the US, Hong Kong and the Netherlands) creating maintainability problems. P5 mentioned *"And where modifications have to be made on several locations because the software is not modularized."* He further detailed that the applications have evolved independently *"[..] sources are copied and modifications are built into the copied files and where that leads to [independent evolution] when a problem appears in regular process and sources are modified because and this modification is necessary in other files as well."*.

Modernization provided a completely new framework and standard ways of developing applications, thereby enhancing maintainability. P4 explained this as *"The programming standards over time have been documented and is now a formal document which is also shared with the vendors, so they start working from that."* He further added: *"What is interesting is that, after the modernization making simple changes has become little easier than before [...] Because people know the code, know the processes, use cases are defined after the modernization so it is easier to maintain."*

3) Increase Usability: With this modernization, the company aimed to improve user-friendliness. P4 highlighted this business goal as *"While in modernization phase we were able to also think of the easiness of working, accessibility of the application, usability of the application."* He expressed that the business needed to be more competitive and indicated that there is a need to have a "smart" looking new application as: *"You wanted to look smart. Business software is always ugly, very annoying to use and that to has become one of my targets when I was managing sales. I was one to have applications which look smart."* P6 stated that *"That was the first decision where modernization took place. I think what influenced modernization is [that] end user experience becomes key in applications."*

During software modernization, the Aviation Co. made architectural changes transforming monolithic legacy code to a layered architecture, including a presentation layer to represent the user interface. P4 expressed *"User experience has become more and more topic. Therefore our decision to come up with multi-layer applications"* He further firmly stated that the usability has increased after modernization by stating *"Modernized system was more user friendly than the old one."*

4) Unintended Benefits: Several unintended benefits were reported by the interviewees. One of them was indirect maintenance cost reduction, which was never their main business goal, yet it was achieved. P5 highlighted this as *"Cost reduction [was] not a business goal. It is a side effect because we have less overhead in maintenance but it's not a business goal to have ICT shrink because we have one source instead of."* Similarly, the company observed increased availability of the new application as compared to the older one. P4 expressed this as *"[...] there is a separate database for intermediate storage and actual data like stock data, part information there is an exchange with core database with messaging files, which makes front end 24/7 available. Independent of availability of the backend system."*

From the organizational perspective, the company has observed organizational flexibility. P5 mentioned that the behavior of the organization has also changed. P6 supported P5's observation as *"Modernization is not only the technical part but it is also the adoption in your organization"*. Similarly, better transparency was also observed in that users were better able to diagnose problems within the new application. P4 commented on this as *"In a later stage we noticed that the new system was working better than the old system and even later stage it was obvious that errors that were reported from the application appeared to be master data errors [rather] than application errors. So, [yes] that was transparent."*

C. Case III: The Public Service Case

The Dutch public service office initiated a project in 2010 to modernize its legacy commercial-off-the-shelf (COTS) application. Due to high maintenance cost and limited vendor support, the public service office decided to modernize the COTS application to an Oracle platform. The legacy application was a heavily customized COTS package and was responsible for managing finances of the office. After successful modernization, the new Oracle-based software system is built upon an enterprise service bus (ESB) that facilitates easy integration of software applications from different government agencies. The project lasted for three years.

1) Reduce Maintenance and Operational Cost: One of the expected benefits is to save cost as mentioned by P7 *"From business perspective they were looking for integration and the cost and time saving of this."* Also from a licensing perspective, the organization was searching for a cheaper option as stated by P7: *"But [new system] was cheaper [..] it will almost likely to be quite [a bit] cheaper because the licenses per user was cheaper in this case. That's why they chose [new system] in first place."*

After modernization, there was a reduction in maintenance and operational cost. This was achieved with significantly lower licensing cost from Oracle as compared to the cost of running [COTS]. The interviewee confirmed that significant savings were achieved after modernization by indicating that the licenses per user was cheaper. The other factor that contributed to cost saving was by reducing number of manpower. P7 mentioned this aspect as *"cost saving, efficiency and number of employees, that if you have a new process that is smoother easier and quicker than you don't need that many employees."*

2) Phase out Legacy Technology: The existing system was a heavily customized [COTS] system, incrementally customized for more than 6 years. With all these customizations, the legacy systems was fit for purpose but required significant efforts to integrate with new systems. P7 expressed this as *"This [COTS] system , existed [for] 6 years or so and is heavily customized. A lot of custom effort to get the way they wanted to do it but it involved lots of customization. The point is that after 6 years later, it was perfect for what they were doing."* Furthermore, the system was not supported by the vendor. P7 explained: *"Well, first of all this was an old system. We were on an older version which was not supported so [legacy system] was to upgrade."* These drawbacks significantly increased cost and hence the organization decided to phase out their legacy technology.

As a result of modernization, the [COTS] system was replaced by an Oracle platform with an ESB. P7 explained the differences by drawing the legacy and the current architectures. In the new architecture, an ESB is used to integrate various applications.

3) Unintended Benefits and Detrimental Effects: As an effect of modernization, the organization has experienced improved organizational flexibility. Some of the geographically separated departments of the organization were merged to one. P7 expressed this as *"Eventually because of this [modernized] system, like I said these departments were geographically separate, but they started working together. But they also moved geographically to one location."* The effect of improved organizational flexibility was also reflected by changes in the process. The interviewee expressed this as *"So there was an updated model in who is allowed to do what within the system especially the process is changed. Then process got integrated and people got different roles within the flow of the system."* Regarding detrimental effects, P7 mentioned that there was some user resistance reported after operationalization of the new system.

D. Case IV: The Gov. Office Case

This is a Government office of Portugal which was running its administration module built in COBOL and DB2, both operating on IBM mainframes. The system was running 24/7 IMS applications to serve users via terminal emulators (directly) or via web-services (indirectly). Prior to modernization, a proof-of-concept was successfully done. Then the modernization project was started in 2010 and involved rehosting and migrating the COBOL application running on an IBM mainframe to a Linux environment, converting code in one COBOL dialect to another, and migrating IMS data to an Oracle database. The modernization project lasted for six months.

1) Reduce Operational Cost: As per available documentation, the primary objective of the modernization was to reduce the maintenance and operational cost due to the use of mainframe technology. One of the initial documents state this as *"ongoing maintenance costs were well above 1M euros/year, and kept increasing."* P8 confirmed that the project was aimed at reducing costs as *"The main purpose was to reduce cost immediately."*

After the successful modernization, the organization made significant cost savings. The organization reported that the maintenance cost was reduced by more than 80%, primarily due to phasing out mainframe systems. P8 reported that *"The costs reduced enormously with [the] same functionality, with more or less same performance at much lower cost."*

2) Phase out Legacy Technology: With this modernization, the organization also wanted to phase out their legacy technology (i.e., the mainframe and COBOL). The key reason for this was the ageing mainframe and COBOL manpower within the organization. P8 mentioned *"Just had two COBOL programmers– one of them was already retired and he would go there part-time to solve problems and to develop some new functions. This was also a concern."*

Due to the strategy taken to modernize the legacy system, this goal was partially realized. The organization opted to first re-host the legacy application from mainframe to Linux but keeping the COBOL within minimal configuration changes. With the cost savings from re-hosting, the organization planned to phase out COBOL in the future. P8 mentioned this as *"we only moved the application out of mainframe and the application was still in COBOL [..]the new system was more flexible and [..] you could develop in other languages like Java and integrate with the application. This was not possible with the mainframe."*

3) Unintended Benefits and Detrimental Effects: It is interesting to observe that there were several unintended (both positive and negative) impacts observed. Several unintended improvements were realized such as improved queries, automatic archiving, improvements to withstand much larger loads than before. Nevertheless, the performance of the new system was below that of the old mainframe system due to the firewall and security mechanisms used in the new Linux environment.

E. Case V: The Finance Co. Case

The Finance Co. is a Dutch financial group. In 2012, the company initiated a modernization project for its payments system built in COBOL and running on an IBM mainframe. The payments application used a DB2 database with approximately 28TB of historical data. CICS was used for transaction monitoring and approximately 10K jobs to run batches. The overall application had 11M LoC for batch programs and 8M LoC for online transaction processing. The modernization process involved re-hosting and migrating the application from the IBM mainframe to an AIX Unix platform, re-developing the existing CICS code for online transactions in Java, migrating DB2 to an Oracle 10 database, and testing. The modernization project lasted for 10 months.

1) Reduce Operational Cost: The Finance Co. had its payments system running on an IBM mainframe and the operational cost, particularly licensing cost of the mainframe-based applications, were steadily increasing. To reduce operational costs, the company started the modernization project. The project manager (P9) worded this as *"The Finance Co. saw the exploitation [operational] costs getting higher and higher and the application itself was limited in its extensibility [..] But the system ran into limitations on the mainframe. They could not grow, well they could, if they invested in expanding the mainframe. But they wanted to cut costs on operating the system."* The project documentation also emphasized "Reduction in operation cost" as one of the main business objectives.

Financial reports after modernization indicate that due to this modernization project, the operational costs have been reduced significantly.

2) Increased Performance: The other key aim of modernization was to increase performance. The processing capacity of the mainframe was reaching its limits within its existing configuration. P9 expressed this as *"The time they needed for the daily process of their mutations was too long. The window they had from 7 pm to 7am [out of office time] started to get close to not being enough. If there was one little problem, the process would need too much time. Once every month, for their big batch, they needed even 60 hours. Concluding, the total lead-time needed every night was too high. A project goal was to lower this time."*

After modernization, the company compared the batch window on the mainframe to that of the AIX Unix environment. The performance gain on the AIX Unix environment was more than a factor of three.

3) Phase out Legacy Technology: The key consideration on achieving cost reduction was by phasing out the mainframe ecosystem. The mainframe ecosystem incurred high licensing cost and the company was approaching the end of the existing contract. The company took this opportunity to discontinue the mainframe. P9 expressed this concern as *"... had the idea already to move to another platform, on the mainframe were some products with high licensing costs. The Finance Co. already researched this, how can we move the applications to Java, but ran into problems."*

Prior to this modernization, the company tried modernizing the COBOL to Java, but that was not successful. Hence, in this project they opted to initially re-host the existing COBOL application from mainframe to AIX Unix and IBM DB2 database to an Oracle database. With this modernization, the company successfully re-hosted their operational environment but did continue to use COBOL.

4) Unintended Benefits: After modernization, the company benefitted from increased flexibility by becoming vendor independent (Mainframe ecosystem) and opened up possibilities to adopt new advanced technologies.

V. FINDINGS

A. Cross-Case Synthesis

Table IV depicts the findings of the cross-case analysis of the five software modernization case studies. In addition to the summary of expected benefits of all five cases, we also list some of the modernization activities that were in fact performed as well as some that we know were not performed. By listing the activities, we aim to provide a high-level view of the commonalities and differences between the five cases in terms of activities. The added value of listing the modernization activities is that they sketch a more detailed picture of the extent of the modernization, and allow us to put the associated benefits better in context. Finally, in the rightmost column of the table we list any side effects that have been observed after modernization. These side effects can be both positive (unintended benefits, marked with $+$), and negative (detrimental effects, marked with $-$).

1) Expected vs. Observed Benefits: As can be observed in Table IV, most of the expected benefits are met after software modernization. The expected benefits include both technical goals such as enhanced usability, increased maintainability, product consolidation, and increased performance as well as business goals such as reducing costs, and phasing out legacy technology. We note that organizational benefits (like organizational flexibility and transparency) that are mentioned mostly as the unintended benefits, are also much less commonly mentioned as claimed benefits in the literature.

In two of the cases (the Gov. Office and Finance Co.), the expected benefit (i.e., *"Phase out legacy technology"*) was only partially met. These two exceptions are interesting, because in both cases this was in fact the main goal of modernization. In the Gov. Office case, phasing out legacy technology was a goal for two reasons: a lack of skilled manpower and the high cost involved in keeping the legacy technology operational. In the Financial Co., the high operational cost was the main reason to phase out legacy technology. In both cases, the choice was made to re-host the legacy COBOL system to different platform. The idea was to reduce operational cost by re-hosting the legacy system on a cheaper platform, in order to save money and thereby create the budget to initiate programming language modernization (porting the COBOL source code to a more modern technology). As indicated in the individual case study, significant cost savings (>60%) were reported. The process of re-hosting the legacy systems to a different platform did require various additional activities: in the case of Finance Co., assembler code was re-written in Java, Rexx scripts were converted to Unix-Rexx, development based on VisualAge was changed to the Enterprise Generation Language that generates Java wrappers. In case of the Gov. Office, the COBOL source code required minor syntactic adaptations, IMS database migration to Oracle database, and the re-development of the libraries.

2) Unintended Benefits: Interestingly, the case companies observed some unintended benefits– benefits that were not originally set down as goals of the modernization. The unintended benefits include recurring goals such as reduced maintenance cost, increased availability, and increased flexibility. Furthermore, three of the cases (Infra Co., Aviation Co. and Public Service) observed organizational benefits in terms of transparency and flexibility. This indicates that a software modernization can bring about significant changes in the organization.

We also observed some interesting opportunities for one of the case companies that arose as a side effect of modernization. The Aviation Co. decomposed their legacy software in a way that allowed them to expose their capabilities as SaaS to one of their customers. Additionally, they used the process of modernization to initiate an improvement to the software development process by introducing programming standards and enforcing quality control checks.

In spite of all these benefits, some detrimental effects were also observed. In two of the cases (Infra. Co. and Public Service) user resistance was reported after modernization. Despite "increase performance" is listed as a claimed benefit (cf. Table I), the opposite was observed for the Gov. Office case: the performance of the new system was lower as compared to the original mainframe system due to additional firewall and security mechanisms on the new platform.

B. Claimed vs. Observed Benefits

This paper provides an opportunity to compare the claimed benefits (cf. Table I) with the observed and unintended benefits (cf. Table IV). Most of these observed and unintended benefits complement and are in–line with the claimed benefits. The observed and unintended benefits form a subset of the claimed benefits of software modernization. However, we do observe some differences. For example, the benefits related to organizational benefits (Organizational flexibility and transparency) attain relatively little attention in the academic literature. On the other hand, the list of claimed benefits contains some benefits reported due to modernizing to specific architectures/-platforms. For instance, modernization to a service-oriented architecture (SOA) promises to deliver software modernization specific benefits such as increased flexibility, reduced costs, increased productivity, increased reusability, faster-time-to market, loose coupling [8], [49], [50]. Since none of our case studies include modernization to SOA, we are not able to assess some of these claims.

C. Lessons Learned

The work reported here is indicative and the sample (case studies) are not large enough to claim comprehensiveness. Despite being an initial research initiative to empirically explore benefits of software modernization, the following lessons can be of interest:

1) **Wider applicability**: Industry can utilize software modernization not only to reduce maintenance cost and to phase out obsolete technology but also for other (business) opportunities. For instance, software modernization can provide an opportunity to redefine the business model of the company. We observed this for Aviation Co. where software modernization enabled the company to offer their modules as services through SaaS. Furthermore, they also used this opportunity to improve their software development process by introducing standards. In the Infra. Co. case, software modernization was used to consolidate their products.
2) **Technical vs. organizational aspects**: Apart from possible technical improvements, software modernization can be used to improve organizational aspects such as bringing transparency and flexibility. This was observed in two of the cases.
3) User resistance was observed in two of the cases. This suggests that any software modernization should also address the soft skill aspects to mitigate such resistances by conducting training and capacity building programs, and by holding seminars to create the necessary awareness about software modernization.
4) In two of the cases (Public service and Finance Co.), a phased approach of mainframe-based software modernization, i.e., initially re-hosting the mainframe-based systems to economical platforms and thereby saving maintenance cost for language modernization in future, was undertaken. This is a worthwhile software modernization alternative that industry can adopt.

VI. VALIDITY

Qualitative research studies are often viewed with discomfort in software engineering [51] with respect to validity as assessing the validity of the qualitative research is a challenging task [52]. Yin [45] argues that the quality of the case studies based on explanatory research should be judged on the basis of the following types of validity:

1) Construct Validity: concerns the validity of the research method and focuses on whether the constructs (i.e., questions, terminology) are interpreted and measured correctly. This is a clear threat to our research as maintaining consistent terminologies and their definitions throughout multiple case

TABLE IV
CROSS-CASE ANALYSIS OF FIVE CASE STUDIES

Case	Modernization Activity	Expected Benefit	Observed Benefit	Other Observation
Infra. Co.	Operational platform change User interface modernization Code optimization	Enhance usability Product consolidation Increase maintainability	Yes Yes Yes	+Organizational transparency +Maintenance cost reduction −User resistance
Aviation Co.	Operational platform change User interface modernization Architectural modernization	Increase flexibility Increase maintainability Enhance usability	Yes Yes Yes	+Maintenance cost reduction +Increased availability +Organizational flexibility +Organizational transparency
Public Service	Architectural modernization Operational platform change Database migration	Phase out legacy technology Reduce maintenance & operational cost Increase flexibility	Yes Yes Yes	+Organizational flexibility −User resistance
Gov. Office	Code conversion Database migration Operational platform change Code analysis	Reduce operational cost Phase out legacy technology	Yes Partially	+Improved queries +Increased load threshold −Decreased system performance
Financial Co.	Operational platform change Code conversion Database migration Code analysis	Reduce operational cost Phase out legacy technology Increase performance	Yes Partially Yes	+Increased flexibility

companies is a challenge. To minimize this threat, we initially provide an interview protocol that includes a brief introduction to the research, interview questions, and an explanation of the terminology used. After transcribing the interviews, the interviewees were contacted by email, if required.

2) External Validity: concerns the domain to which the results can be generalized. With respect to external validity, we do not claim comprehensiveness of the finding; it is rather indicative. The diversity of the case companies in terms of domain, company size and geography gives confidence that the findings represent a significant view. Furthermore, the cross-case analysis method that we adopted to synthesize the findings arguably increases the generalizability. However, more empirical studies will have to be designed and executed to extend the validity of the findings. In particular, when we compare the claimed benefits from the literature (see Table I) with the expected and unintended benefits of the five cases, then it is clear that we do not cover them all. In particular, among the claimed benefits we also find benefits reported due to modernizing to specific architectures/platforms. For instance, modernization to a service-oriented architecture (SOA) promises to deliver software modernization specific benefits like increased flexibility, reduced costs, increased productivity [8] along with SOA specific benefits such as increased reusability, faster-time-to market, loose coupling, statelessness [49], [50]. Since none of our cases involved modernization to SOA, we are not able to assess these claims.

3) Reliability: is concerned with demonstrating that the results of the study can be replicated. This threat is mitigated by maintaining a case study database[3] that contains all the relevant information used in the case study. This case study database consists of anonymized interview transcripts, Nvivo coding, interview protocols and backward snowballing. We believe that these artifacts contribute towards the transparency.

A few remarks should be made on the retrospective nature of the case study. A problem relating to this type of study is hindsight bias– a belief that an event is more predictable after it becomes known than it was before it became known [53]. In this research, it means that interviewees might tend to reconstruct the business goals based on the results of the impacts. We minimized hindsight bias by using multiple interviewees within same company and documentation, whenever possible.

[3]Available at https://servicifi.wordpress.com/ICSME-2

VII. CONCLUSION

Research on software modernization suggests many "claimed" benefits of modernization with limited empirical evidences to support the claims. A plethora of technical solutions of software modernization are labeled as "successful" once the modernization process is proven to be technically feasible. There is limited research that assesses whether these technically "successful" solutions do meet the expected benefits, i.e., the pre-modernization business goals.

In this paper, we address this gap of empirical evidence by discussing five (retrospective) case studies of software modernization. We have documented the "expected benefits" of each case and considered whether these "expected benefits" were in fact met after modernization. In general, the outcome of these five case studies suggest that the "expected benefits" were observed after modernization. Interestingly, we found that the case companies also observed several "side effects" of modernization: most of them were "unintended benefits", but also a few detrimental effects, primarily, due to user resistance and decreased performance. Among the reported unintended benefits we also found benefits that received relatively little attention in software modernization literature, in particular, organizational transparency and organizational flexibility.

To summarize, this paper has the following contributions:

- reports upon five retrospective modernization case studies within different organizations,
- documents the pre-modernization business goals as "expected benefits" and identifies whether these benefits were in fact met, and provides a comparative analysis of "claimed" and "expected" benefits of modernization.

To the best of our knowledge, this empirical research is the first to explore and validate the observable benefits of software modernization. It is clear that more empirical studies have to be performed in collaboration with industry to further extend and strengthen the findings. In particular, when we compare the claimed benefits from the literature to the expected benefits of our case studies, then some benefits are still missing. Furthermore, study of successful case studies about modernization to a specific architecture/platform (e.g., legacy to SOA or cloud, code conversion/transformation) can provide insights into more specific benefits. In case of software modernization, we believe that an empirical study of failure cases is invaluable to fully understand modernization impacts, making this an important topic for future work.

References

[1] L. A. Belady and M. M. Lehman, "A model of large program development," *IBM Sys. J.*, vol. 15, no. 3, pp. 225–252, 1976.

[2] V. Rajlich, "Software evolution and maintenance," in *FOSE*. ACM, 2014, pp. 133–144.

[3] T. Mens, Y.-G. Gueheneuc, J. Fernandez-Ramil, and M. D'Hondt, "Guest editors' introduction: Software evolution," *IEEE Soft.*, vol. 27, no. 4, pp. 22–25, 2010.

[4] K. H. Bennett and V. T. Rajlich, "Software maintenance and evolution: a roadmap," in *FOSE*. ACM, 2000, pp. 73–87.

[5] K. Bennett, "Legacy systems: Coping with success," *IEEE Soft.*, vol. 12, no. 1, pp. 19–23, 1995.

[6] S. Comella-Dorda, K. Wallnau, R. C. Seacord, and J. Robert, "A survey of black-box modernization approaches for information systems," in *ICSM*. IEEE, 2000, pp. 173–183.

[7] R. Khadka, G. Reijnders, A. Saeidi, S. Jansen, and J. Hage, "A method engineering based legacy to soa migration method," in *ICSM*. IEEE, 2011, pp. 163–172.

[8] K. A. Nasr, H.-G. Gross, and A. van Deursen, "Realizing service migration in industrylessons learned," *JSEP*, vol. 25, no. 6, pp. 639–661, 2013.

[9] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Emp. Soft. Eng.*, vol. 14, no. 2, pp. 131–164, 2009.

[10] M. W. Godfrey and D. M. German, "The past, present, and future of software evolution," in *FoSM*. IEEE, 2008, pp. 129–138.

[11] R. Khadka, B. V. Batlajery, A. M. Saeidi, S. Jansen, and J. Hage, "How do professionals perceive legacy systems and software modernization?" in *ICSE*. ACM, 2014, pp. 36–47.

[12] H. Gall, M. Jazayeri, R. R. Klosch, and G. Trausmuth, "Software evolution observations based on product release history," in *ICSM*. IEEE, 1997, pp. 160–166.

[13] M. W. Godfrey and Q. Tu, "Evolution in open source software: A case study," in *ICSM*. IEEE, 2000, pp. 131–142.

[14] S. Koch, "Software evolution in open source projects–a large-scale investigation," *JSME*, vol. 19, no. 6, pp. 361–382, 2007.

[15] C. F. Kemerer and S. Slaughter, "An empirical approach to studying software evolution," *TSE*, vol. 25, no. 4, pp. 493–509, 1999.

[16] H. Kagdi, M. L. Collard, and J. I. Maletic, "A survey and taxonomy of approaches for mining software repositories in the context of software evolution," *JSME*, vol. 19, no. 2, pp. 77–131, 2007.

[17] J. Fernandez-Ramil, A. Lozano, M. Wermelinger, and A. Capiluppi, "Empirical studies of open source evolution," in *Soft. Evol.* Springer, 2008, pp. 263–288.

[18] R. Robbes and M. Lanza, "A change-based approach to software evolution," *Elec. Notes in TCS*, vol. 166, pp. 93–109, 2007.

[19] M. Lanza, "The evolution matrix: Recovering software evolution using software visualization techniques," in *IWPSE*. ACM, 2001, pp. 37–42.

[20] I. D. Baxter, C. Pidgeon, and M. Mehlich, "Dms®: Program transformations for practical scalable software evolution," in *ICSE*. IEEE Computer Society, 2004, pp. 625–634.

[21] J. Singer, "Practices of software maintenance," in *ICSM*. IEEE, 1998, pp. 139–145.

[22] A. Bianchi, D. Caivano, F. Lanubile, F. Rago, and G. Visaggio, "An empirical study of distributed software maintenance," in *ICSM*. IEEE, 2002, pp. 103–109.

[23] M. Razavian and P. Lago, "A frame of reference for SOA migration," in *Towards a Service-Based Internet*. Springer, 2010, pp. 150–162.

[24] A. Almonaies, J. Cordy, and T. Dean, "Legacy system evolution towards Service-Oriented Architecture," in *SOAME'10*. IEEE, 2010, pp. 53–62.

[25] P. Jamshidi, A. Ahmad, and C. Pahl, "Cloud migration research: A systematic review," *Tran. Cloud Comp.*, vol. 1, no. 2, pp. 142–157, 2013.

[26] S. Jalali and C. Wohlin, "Systematic literature studies: database searches vs. backward snowballing," in *ESEM*. ACM, 2012, pp. 29–38.

[27] H. Sneed, "Planning the reengineering of legacy systems," *IEEE Soft.*, vol. 12, no. 1, pp. 24–34, 1995.

[28] H. C. Benestad, B. Anda, and E. Arisholm, "Understanding software maintenance and evolution by analyzing individual changes: a literature review," *JSME*, vol. 21, no. 6, pp. 349–378, 2009.

[29] J. Bisbal, D. Lawless, B. Wu, and J. Grimson, "Legacy information systems: Issues and directions," *IEE Soft.*, vol. 16, no. 5, pp. 103–111, 1999.

[30] A. Erradi, S. Anand, and N. Kulkarni, "Evaluation of strategies for integrating legacy applications as services in a service oriented architecture," in *SCC*. IEEE, 2006, pp. 257–260.

[31] M. L. Brodie, "The promise of distributed computing and the challenges of legacy systems," in *Advanced Database Systems*. Springer, 1992, pp. 1–28.

[32] G. Canfora, A. Cimitile, A. De Lucia, and G. A. Di Lucca, "Decomposing legacy programs: A first step towards migrating to client–server platforms," *JSS*, vol. 54, no. 2, pp. 99–110, 2000.

[33] H. Gall, R. Klösch, and R. Mittermeir, "Object-oriented re-architecturing," in *ESEC*. Springer, 1995, pp. 499–519.

[34] J. Koskinen, J. J. Ahonen, H. Sivula, T. Tilus, H. Lintinen, and I. Kankaanpaa, "Software modernization decision criteria: An empirical study," in *CSMR*. IEEE, 2005, pp. 324–331.

[35] J. Bisbal, D. Lawless, B. Wu, J. Grimson, V. Wade, R. Richardson, and D. O'Sullivan, "An overview of legacy information system migration," in *APSEC/ICSC*. IEEE, 1997, pp. 529–530.

[36] M. L. Brodie and M. Stonebraker, "DARWIN: On the incremental migration of legacy information systems," GTE Labs Inc, TR TR-022-10-92-165, 1993.

[37] B. Wu, D. Lawless, and e. a. Bisbal, "The butterfly methodology: A gateway-free approach for migrating legacy information systems," in *ICECCS*. IEEE, 1997, pp. 200–205.

[38] H. M. Sneed, "Software renewal: A case study," *IEEE Soft.*, vol. 1, no. 3, pp. 56–63, 1984.

[39] A. van Deursen, P. Klint, and C. Verhoef, "Research issues in the renovation of legacy systems," in *FASE*. Springer, 1999, pp. 1–21.

[40] A. Mehta and G. T. Heineman, "Evolving legacy system features into fine-grained components," in *ICSE*. ACM, 2002, pp. 417–427.

[41] W. S. Adolph, "Cash cow in the tar pit: Reengineering a legacy system," *IEEE Soft.*, vol. 13, no. 3, pp. 41–47, 1996.

[42] M. Mortensen, S. Ghosh, and J. M. Bieman, "Aspect-oriented refactoring of legacy applications: An evaluation," *TSE*, vol. 38, no. 1, pp. 118–140, 2012.

[43] A. De Lucia, R. Francese, G. Scanniello, and G. Tortora, "Developing legacy system migration methods and tools for technology transfer," *SPE*, vol. 38, no. 13, pp. 1333–1364, 2008.

[44] G. Lewis, E. Morris, and D. Smith, "Service-oriented migration and reuse technique (SMART)," in *STEP*. IEEE, 2005, pp. 222–229.

[45] R. Yin, *Case study research: Design and methods*. Sage, 2009.

[46] B. Flyvbjerg, "Five misunderstandings about case-study research," *Qualitative inquiry*, vol. 12, no. 2, pp. 219–245, 2006.

[47] C. Seaman, "Qualitative methods in empirical studies of software engineering," *TSE*, vol. 25, no. 4, pp. 557–572, 1999.

[48] K. M. Eisenhardt, "Building theories from case study research," *Academy of management review*, vol. 14, no. 4, pp. 532–550, 1989.

[49] T. Erl, *Service-oriented architecture: concepts, technology, and design*. Pearson Education India, 2006.

[50] M. Papazoglou, *Web services: principles and technology*. Pearson Education, 2008.

[51] S. Adolph, W. Hall, and P. Kruchten, "Using grounded theory to study the experience of software development," *Emp. Sof. Eng.*, vol. 16, no. 4, pp. 487–513, 2011.

[52] N. Golafshani, "Understanding reliability and validity in qualitative research," *The Qualitative Report*, vol. 8, no. 4, pp. 597–607, 2003.

[53] N. J. Roese and K. D. Vohs, "Hindsight bias," *Perspectives on Psychological Science*, vol. 7, no. 5, pp. 411–426, 2012.