The Journal of Systems and Software 164 (2020) 110548

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Leading successful government-academia collaborations using FLOSS and agile values☆

Melissa Wen[a,*], Rodrigo Siqueira[a], Nelson Lago[a], Diego Camarinha[a], Antonio Terceiro[c], Fabio Kon[a], Paulo Meirelles[b,a]

[a] University of São Paulo – IME-USP, Brazil
[b] Federal University of São Paulo – UNIFESP, Brazil
[c] Linaro Limited, Brazil

ARTICLE INFO

Article history:
Received 15 November 2018
Revised 17 January 2020
Accepted 11 February 2020
Available online 11 February 2020

Keywords:
Project management
Government-Academia collaboration
Free software
Open source software
Agile methodologies
e-Government

ABSTRACT

Government and academia share concerns for efficiently and effectively servicing societal demands, which includes the development of e-government software. Government-academia partnerships can be a valuable approach for improving productivity in achieving these goals. However, governmental and academic institutions tend to have very different agendas and organizational and managerial structures, which can hinder the success of such collaborative projects. In order to identify effective approaches to overcome collaboration barriers, we systematically studied the case of the Brazilian Public Software portal project, a 30-month government-academia collaboration that, using Free/Libre/Open Source Software practices and agile methods for project management, developed an unprecedented platform in the context of the Brazilian government. We gathered information from experience reports and data collection from repositories and interviews to derive a collection of practices that contributed to the success of the collaboration. In this paper, we describe how the data analysis led to the identification of a set of three high-level decisions supported by the adoption of nine best practices that improved the project performance and enabled professional training of the whole team.

1. Introduction

The use of information and communication technologies (ICT) to support government procedures and routines, engage citizens, and provide government services (Scholl, 2003), known as Electronic Government (e-government), has become synonymous with the idea of a modern state. While "e-government" has different meanings for different people (Chandra Misra, 2006), such projects virtually always differ from most other ICT projects due to their complexity and large size (Anthopoulos et al., 2016). On the one hand, they are complex because they combine innovation, information & communications technologies, politics, and social impact. On the other hand, they are large in their scope, target audience, organizational size, and time.

The development of innovative e-government projects that meet the needs of society may be addressed through collaboration between government and academia. However, managing such collaborative projects is often challenging. Indeed, one of the leading causes of e-government project failure is poor project management (Anthopoulos et al., 2016) caused by difficulties such as lack of creativity and vision, poor communication and organization skills, unclear work-breakdown, ineffective workload management, poor scope definition, and change management, shifting requirements, technical complexity, and poor delegation and tracking. In a joint project between government and academia, other specific challenges become significant, such as organizing the collaboration around the project, aligning goals, synchronizing the pace of both partners, and overcoming the failure trend of e-government projects (Goldfinch, 2007). Therefore, proper management of the collaborative project should be a relevant concern when government and academia combine efforts to develop an e-government solution.

In the context of software development projects, an institution adopts development methods that best meet its managerial procedures and organizational culture. For example, academia can work on cutting-edge development methodologies while the gov-

☆ The 30-month case of rebuilding the Brazilian Public Software portal

* Corresponding author.
E-mail addresses: wen@ime.usp.br (M. Wen), siqueira@ime.usp.br (R. Siqueira), lago@ime.usp.br (N. Lago), diegoamc@ime.usp.br (D. Camarinha), antonio.terceiro@linaro.org (A. Terceiro), kon@ime.usp.br (F. Kon), paulo.meirelles@unifesp.br (P. Meirelles).

https://doi.org/10.1016/j.jss.2020.110548
0164-1212/

ernment typically relies on traditional techniques. When two large-scale, complex organizations decide to develop a solution collaboratively, the development methods and workflow of one may conflict with the culture and interests of the other. Changing the development process of either of them represents an organizational disturbance with impacts on structure, culture, and management practices (Nerur et al., 2005). Accordingly, when in collaboration, government and academia should strive to increase the chances of success, given the typical tight deadlines and limited outlay of such projects. Thus, they should avoid interferences in their respective managerial processes that result only in negative impacts on the project schedule and in budget wastage. Actual mechanisms to achieve this, however, are not straightforward.

Difficulties such as these are part of the landscape of many FLOSS (Free-Libre-Open Source Software) projects. The development of such projects is frequently a collaborative work involving multiple institutions with different interests and organization without their internal processes being affected. The significant success of FLOSS, which has considerable overlap with agile methodologies, to deliver complex and quality products, reinforces the power of applying their values and practices in collaborative scenarios. Understanding how to overcome conflicting differences between crucial stakeholders could lead a collaboration to success, especially in the case of government and academia.

Concerned with understanding the best practices that government and academia can adopt to develop high-quality, well-structured, resource-saving e-government solutions jointly, we focus on the following two research questions:

RQ1. *How to introduce FLOSS and agile best practices into government-academia collaboration projects?*

RQ2. *What FLOSS and agile practices favor effective team management in government-academia collaborative projects?*

We explore these questions based on a case study about the SPB (Brazilian Public Software, e.g., Sofware Público Brasileiro, in Portuguese). The project behind it was an unprecedented 30-month software development structured as a partnership between an academic team and a government agency aiming at three goals: the development of a novel product (the SPB Portal); the acquisition of experience and expertise regarding agile and FLOSS best practices within the governmental agency; and the professional and educational evolution of the involved undergraduate students. As we will see, this project exhibited many of the complicating characteristics of government-academia co-development projects. It enabled us to showcase both quantitative and qualitative analyses of the benefits of FLOSS and agile practices.

To this end, we identified the applied best practices from FLOSS ecosystems and agile methodologies used in the project. We collected and analyzed data from the project repository and conducted a survey targeted at the project participants to extract their perception about how these practices were useful for government-academia collaboration. From this data, we identified three high-level project management decisions that improved the project performance and contributed to the success of the partnership: (1) the use of the system under development to develop the system itself; (2) bringing together the government staff and the development team; (3) organizing the development team into priority fronts, and for each one, hire at least one specialist from the IT market. Starting from these decisions and the perceived benefits brought by them, we proceed to map how actual day-to-day practices materialized them. By identifying them, we aim to help academia to understand better critical issues they will be confronted with when engaging in a government-academia software project.

This paper is an extension to previous studies from our research group concerning a Brazilian government-academia collaboration project. The first study about the SPB portal development project (Meirelles et al., 2017) was an experience report that revealed lessons learned by the academic members responsible for coordinating project activities (professors and senior developers). In the second, Siqueira et al. (2018) focused on presenting its continuous delivery approach, technically describing the pipeline created to support large organizations in the development of a system of systems. In a more recent conference paper, Wen et al. (2018) examined the method developed over the 30 months of the project using both agile values and standards from the FLOSS community to mitigate the cultural gap between organizations.

Here we provide a comprehensive view of our previous studies on government-academia collaboration. We go beyond the earlier works by systematically investigating and describing how the practices adopted favor a government-academia collaboration. Moreover, we discuss reflections from a complementary questionnaire sent to participants and provide critical analysis of the benefits and consequences of the decisions and the adopted practices. The novel research resources enabled us to: (i) find new insights on software engineering training; (ii) reach additional benefits that improve collaboration; and (iii) identify how the experience with the adopted practices has been reflected in the participants' professional skills a few years after the end of the project.

The paper is organized as follows. Section 2 addresses underlying concepts and research opportunities related to this study, such as co-development, knowledge transfer, and adoption of management practices in both large organizations and public administration. We discuss the characteristics and associated difficulties of collaborative product development, especially academia-government partnerships and e-government endeavors, and how FLOSS and agile methods may be suitable in such scenarios. Section 3 presents the object of our case study: the SPB project. We describe its origins and goals, the components that comprise it, and the teams and organizations involved. In Section 4, we state our research questions, focused on the use of FLOSS and agile techniques to support government-academia collaboration, provide a more detailed overview of the SPB project, and describe how we collected and organized the analyzed data, which culminated in the identification of the decisions, practises, and benefits that constitute the core of this work. Section 5 offers a systematic view of the obtained results, detailing the high-level project management decisions and the adopted practices of the SPB project as well as mapping their benefits to the collaboration as a whole, and discusses their consequences to the involved people. In Section 6, we discuss how this research coincides with and differs from previous works, presenting the results reported in the related literature and contrasting them with our findings. Finally, in Section 7, we summarize the material, highlighting its main contributions, pointing out some of its limitations, and suggesting paths to future works.

2. Background

As mentioned above, this paper describes a partnership between an academic team and a government agency with multiple, disparate goals. Accordingly, we see this partnership not as a simple customer-supplier relationship, but as a collaborative product development process, with "two or more partners joining complementary resource and experience with mutual aims, to design or develop a new or improved product" (Büyüközkan and Arsenyan, 2012). Such partnerships involve additional management risks and challenges beyond those posed by product development itself (Littler et al., 1995). Therefore, this scenario demands special attention and leads to interesting research opportunities.

Software co-development is a kind of collaborative product development (Chesbrough and Schwartz, 2007; Büyüközkan and Arsenyan, 2012) concerning different models of software development collaborations (Kourtesis et al., 2012). The relation-

ship between co-development, platforms, and ecosystems has evolved with the advent of cloud computing (Kourtesis et al., 2012). Large-scale software products evolved to platforms for co-development and software ecosystems, with central coordination for software development (Kourtesis et al., 2012). According to Kourtesis et al. (2012), the advantages of this approach are "decreased software and business development costs, quicker time-to-market, improved focus, reduced complexity, and economic profit". They include FLOSS projects as examples of software platforms open for all involved partners, which is the focus of our study.

Complex and large-scale organizations, such as the public administration, have to deal with multiple project variables. In public administration, software products are acquired or developed in a rigid framework of contract obligations. This factor has helped to increase "complexities, delays or even failed delivery of digital services" (Mergel, 2016). As one of the responses for that, government agencies are adopting agile methods (Balter, 2011; Margetts and Dunleavy, 2013) "to update large-scale legacy systems and adapt to environmental changes and citizen requests faster" (Mergel, 2016).

A relevant aspect we discuss in this paper is the adoption of agile practices. Nerur et al. (2005) recognized critical issues concerning the migration from traditional to agile software development by comparing practices of both methodologies. The authors point out managerial, organizational, people, process, and technological issues to be rethought and reconfigured in an organization for a successful migration. They concluded that agile methodologies are more suitable in projects with a high variation of requirements, technical capacities, and technologies. Formal and bureaucratic organizations have more difficulty in the adoption of such methods.

Agile methods represent a significant departure from traditional software development and rely on a different management paradigm. Strode et al. (2009) investigated the relationship between the adoption of agile methodologies and organizational culture by evaluating nine projects. They identified a set of six factors (feedback and learning, teamwork, empowerment of people, focus on results, innovative leadership, and mutual trust) directly linked to agile methods. They concluded that the presence of these aspects in an organization correlates with the effective use of agile methodologies in their projects.

In spite of the growing knowledge about agile methods, they are not always easy to implement. Melo et al. (2013) investigate the growing adoption of agile methods in the Brazilian IT industry. The results of their survey highlight some mismatch that companies face when developing software for public administrations. They show that the acceptance of agile methods has changed in the last decades. The software development industry claims to follow some of the recommendations of the agile manifesto, but some universities and companies are still resistant to adopt such methods.

Beyond the reasonably well-known agile methods, we also highlight here the relevance of FLOSS values and practices. Yet, they are not so well defined. Some studies tried to identify FLOSS practices, while others attempted to determine the relationship between FLOSS practices and agile methods. Raymond (1999), in a seminal essay, described FLOSS best practices from observations of the Linux kernel project and also reflections of his experience with the FLOSS communities. Capiluppi et al. (2003) examined about 400 projects to find FLOSS project properties. In their work, they extracted generic characterization (project size, age, license, and programming language), analyzed the average number of people involved in the project, the community of users, and documentation characteristics. Warsta and Abrahamsson (2003) found differences and similarities between agile development and FLOSS practices. The authors argued that FLOSS development might differ from agile in their philosophical and economic perspectives; on the other hand, both approaches share the definition of work. In its turn, Fraser et al. (2006) claimed that FLOSS is a kind of Agile software development methodology.

Discussions about the relationship between FLOSS and Agile methods are frequently concerned with managing the software project (Javdani Gandomani et al., 2013). Okoli and Carillo (2012) explained this relationship by comparing the characteristics of FLOSS and Agile approaches. While Magdaleno et al. (2012) stated that the relationship between FLOSS and Agile practice was still embryonic at the time of their study (2012), others point in the same direction as our work, reporting that FLOSS and Agile share similar values (Adams and Capiluppi, 2009; Tsirakidis et al., 2009; Corbucci and Goldman, 2010; Javdani Gandomani et al., 2013). For instance, both approaches rely on self-organized teams as well as on team-wide shared and coherent goals (Adams and Capiluppi, 2009; Tsirakidis et al., 2009). Several studies also pointed out that FLOSS and Agile methods support each other in some practices and in tracking the progress of the software development project (Porruvecchio et al., 2007; Deshpande and Riehle, 2008; Adams and Capiluppi, 2009; Lavazza et al., 2010; Gary et al., 2011; Okoli and Carillo, 2012; Rahman et al., 2018). Moreover, Düring (2006) and Goth (2007) describe the use of FLOSS and Agile methodologies simultaneously, which is the same approach analyzed in our research.

Nevertheless, none of these works provide enough evidence about the comprehensive integration of FLOSS and Agile when used simultaneously in a project (Javdani Gandomani et al., 2013). Even recent works such as Harzl (2017) and Müller (2018), who present how FLOSS projects can apply agile management frameworks to favor software development planning, do not advance this issue, especially in large-scale and complex cases such as ours.

In light of this scenario, our work presents a government-academia collaboration for co-developing a production-level solution. We analyzed the decisions made during the life cycle of a real project from the FLOSS and agile perspectives. Our findings are in line with the idea of "continuous activities", which together is part of the roadmap for continuous software engineering proposed by Fitzgerald and Stol (2017). Based on questionnaires, interviews, and development activities data, we extracted the (continuous) best practices that helped to harmonize the interactions between two different development processes, as well as satisfied the management processes of government and academia sides.

3. The case of the Brazilian Public Software portal

The Brazilian government designed the original Brazilian Public Software (SPB) portal in 2005 and released it in 2007. It is a web system that has consolidated itself as an environment for sharing software projects (Freitas and Meffe, 2008). The SPB portal[1] also provides a space (community) for each software. These communities have tools to promote collaboration and interaction among managers, users, and developers, according to practices FLOSS communities use. It was created to support the plan of the government to foster public adoption of FLOSS solutions.

Initially, the portal's purpose was only to share software developed for or by any of the many Brazilian government branches to reduce the costs of hiring software in others. However, for every new software project released, a community was formed around it. Several people, then, started collaborating and sharing the results obtained through the use of those solutions, as commonly occurs in FLOSS (Ducheneaut, 2005).

The concept of a Brazilian Public Software goes beyond FLOSS (Freitas and Meffe, 2008). In addition to being licensed under a FLOSS license, the software needs to have explicit guarantees

[1] https://pt.wikipedia.org/wiki/Software_livre_nos_governos#Software_Público_Brasileiro_(SPB).

that it is a public good and its project must be available on the SPB portal. Inclusion in the SPB Portal also has extra requirements, such as having a public version control system, installation manual and hardware requirements specification (Meirelles et al., 2017).

Notwithstanding its value, in practice, in 2009 the SPB Portal started having several technical issues. The original codebase development stopped, and the SPB Portal did not receive any code updates. To recover the SPB portal, in January 2014, the University of Brasília (UnB) and the University of São Paulo (USP), in a partnership with the Brazilian Ministry of Planning, Budget, and Management (MPOG), designed a project to evolve the SPB portal into an integrated platform for collaborative software development (CDE) (Booch and Brown, 2003) with additional social capabilities. The new SPB Portal was developed by partially distributed teams of undergraduate interns, IT professionals and professors from the Advanced Laboratory of Production, Research, and Innovation in Software Engineering (LAPPIS/UnB[2]) and the FLOSS Competence Center at USP (CCSL/USP[3]), both with experience in FLOSS development.

The new SPB portal[4] includes features such as social networking, mailing lists, version control system, and source code quality monitoring. As of October 2018, it had 83 software communities and 7347 user accounts, mostly from government employees. It is a novelty in the context of the Brazilian government, due to the technologies employed and its various features. Moreover, these characteristics led the project to interact with different FLOSS projects and communities. At the end of the project, the teams built a system-of-systems (Nielsen et al., 2015), adapting and integrating five existing FLOSS projects:

Colab [5] is a system integration platform for web applications. It provides a unified view and interface for a set of different software systems. Thus, users do not notice significant differences when they shift their interaction from one application to the other. For that, Colab provides facilities for (i) centralized authentication, (ii) visual consistency, (iii) relaying of events between applications and (iv) an integrated search engine. Colab implements this integration by working as a reverse proxy for the applications, i.e., all external requests pass through Colab before reaching them.

Noosfero [6] is a software for building social and collaborative networks. Besides the classical social networking features such as uploading content and enabling commentaries to make conversations, it also provides publication features such as blogs and a general-purpose CMS. Most of the user interactions with SPB is through Noosfero: user registration, project and documentation, and contact forms.

GitLab [7] is a web-based Git repository manager with wiki pages and issues tracking features. It is a FLOSS platform and focuses on delivering a holistic solution that will support developers from idea to production seamlessly and on a single platform. GitLab has several unique features, such as built-in continuous integration and continuous deployment, flexible permissions, tracking of Work-in-Progress work, moving issues between projects, group-level milestones, creating new branches from issues, dashboard and time tracking.

Mezuro [8] is a platform to collect source code metrics to monitor the internal quality of software written in C, C++, Java, Python, Ruby, and PHP. It provides a single interface grouping available metric collection tools, allows the selection and composition of metrics flexibly, stores the metrics evolution history, presents results in a friendly way, as well as allows users to customize the given metric values interpretation according to their context.

GNU Mailman [9] is an application for managing electronic mail discussion and e-newsletter lists. Colab itself provides a web interface for GNU Mailman to allow the dialogue and communication between developers, users, and enthusiasts of a determined software. Each software community has its mailing list whose privacy settings can be configured by the software community administrators.

All these integrated systems involve a total of 106,253 commits and 1,347,421 lines of code. In summary, Colab orchestrates these multiple systems using a plugin architecture and smoothly provides a unified interface to final users, including single sign-on and global searches. Without it, the integration of the several features of these systems and other needed back-end features written in different programming languages and frameworks would require a non-trivial amount of work.

In this scenario, we studied the open and collaborative software development practices that inspired the SPB project progress. The academic teams empirically defined an adaptation of different agile and FLOSS communities practices to guide the development process, with a high degree of automation resulting from DevOps practices and also a working process executed in a cadenced and continuous way with the government team.

While the development itself was handled by the academic team, this was not a typical outsourcing scenario:

- The academic partner had a voice in the strategic decisions, as well as the requirements and features definitions;
- Students in leadership roles participated in meetings with government managers;
- Much of the communication and validation of the project converged to the GitLab tool within the project itself;
- The group created a “translation” mechanism from the development team project assessment to the government’s managerial tools;
- Budget management was a concern for both partners;
- The adopted Continuous Delivery pipeline highlights the collaborative aspect of the project workflow;
- One of the goals of the partnership was to promote the transference of knowledge from academia to the government staff regarding FLOSS and agile practices.

In summary, during the SPB project, the academia and the government coordinated and divided the responsibilities of the management activities, sharing concerns about funding and accountability, and business rules and strategic decisions.

4. Research design

A government-academia collaboration project demands strong inter-institutional management, which involves financial issues, requirements and priorities definitions, and possible divergences in the methodologies and development techniques to be employed. In most cases, this kind of software development partnership sets up work teams with different experiences, knowledge, maturity, and work performances. These managerial and organizational complexities resemble the peculiarities found in FLOSS ecosystems, which also often involve multiple institutions with different interests and backgrounds.

With this in mind, FLOSS and agile practices may adequately handle differences in the pace between the institutions involved, improving the cooperation of distinct teams and benefiting the

[2] https://lappis.rocks.
[3] http://ccsl.ime.usp.br.
[4] https://softwarepublico.gov.br.
[5] https://github.com/colab
[6] https://gitlab.com/noosfero/noosfero
[7] https://gitlab.com
[8] https://mezuro.github.io
[9] https://list.org

software being developed. From the FLOSS ecosystems, we can extract useful practices such as open communication, project modularity, development of a community of users, and fast response to reported issues (Capiluppi et al., 2003; Warsta and Abrahamsson, 2003). Besides, individuals and interactions, working software, customer collaboration, and responsiveness to change (Beck et al., 2010) are agile values also suitable in this context.

4.1. Research questions

As mentioned before, we are interested in handling the different development processes at play in collaborative projects, primarily in government-academia joint ventures. To guide this discussion, we focus on two research questions:

RQ1. *How to introduce FLOSS and agile best practices into government-academia collaboration projects?*

We begin by examining 30 months of a government-academia collaboration project to identify managerial and organizational changes. This partnership enabled the introduction of agile methods and FLOSS practices such as collaborative environment development, use of a version control tool and discussion list, continuous delivery, and self-organization. Our study aims to reveal ways to bring such practices into the development process without sacrificing or causing significant impacts on the internal procedures of the institutions involved.

We focus our analysis on issues related to organizational differences and diversity of project members regarding maturity and experience in collaborative development. The harmony between teams sought not only to approximate the mindset and work culture but also to circumscribe the interactions between different roles and responsibilities.

RQ2. *What FLOSS and agile practices favor effective team management in government-academia collaborative projects?*

We continue by pinpointing how the introduction of FLOSS and agile methods has improved interactions between the institutions as well as the internal development teams. For this, we gathered opinions through interviews and questionnaires from three different roles of the project: senior developers, interns, and government staff. In addition to these responses, we also analyze data documented in the project management and development platform to capture the main benefits of the changes in the collaboration management and development model.

4.2. Research method

Due to the exploratory and explanatory character of RQ1 and RQ2, we used the case study as the research method to answer these questions. This method is a good fit when asking “how” or “why” questions and when we can directly observe the events under study and interview persons involved in the activities (two sources of evidence relevant for case studies) (Yin, 2009). Nonetheless, we have also analyzed repository data to ratify our results.

To conduct the data analysis, we adopted *coding* (Charmaz, 2008), the core strategy of Grounded Theory (Glaser and Strauss, 1999; Corbin and Strauss, 2014). It is a research approach widely-used in software engineering (Waterman et al., 2015; Stol et al., 2016; Santos et al., 2016; Hoda and Noble, 2017). We describe in detail our data analysis strategy in Section 4.2.4.

Fig. 1 summarizes our investigation workflow, which had as its starting point previous works about the SPB project (Meirelles et al., 2017; Siqueira et al., 2018; Wen et al., 2018). The rectangles identify our research approaches, and the four following subsections describe how we used each one. The light blue ellipses represent our findings through the methods used. We explain and discuss them in Section 5.

4.2.1. Case study

A case study is the exhaustive study of a single case with the purpose of enlightening a broader class of cases (Gerring, 2006). According to Yin (2009), the essence of a case study is to shed light on a decision or a set of decisions, trying to understand the reasons and ways for their implementation and the lessons learned. With this in mind, we chose the SPB Portal as a case study because, besides being a high-quality example of a well-balanced government-academia managerial process, it offered a valuable opportunity to explore the benefits and challenges of using FLOSS (Kon et al., 2011; DeKoenigsberg, 2008; Fagerholm et al., 2013; 2014) and Agile (Steghöfer et al., 2016; Harzl, 2017; Müller, 2018) practices for Software Engineering due to the diversity of actors involved and the relationships between government, academia, and industry.

On the academic side, the project team was composed of 42 undergraduate interns, two college professors, six developers with significant experience in the FLOSS tools integrated into the portal, and two UX specialists from the IT market. On the government side, the project was managed by two requirement analysts and a board of directors composed of one department director and one department coordinator. While government staff was more used to traditional project management approaches, the academic members believed that the use of agile methods was more appropriate for the development of the proposed platform. As a consequence, conflicts between the internal management processes and differences in pace and goals of each institution started to compromise the platform development. Moreover, during the project, the government board of directors changed and, with it, the vision of the project, affecting the previously approved project requirements. Under these circumstances, the professors who coordinated the project made decisions in a non-systematic way, incrementally adopting best practices derived from FLOSS and agile values to improve the project management process. Finally, during the six initial months, the project received only a fraction of the planned funding for the period, pushing the coordinators to restrict work to formatting the process to be used by the project, in particular through training undergraduate interns on the relevant tools. Much of the project activity from this period focused on requirements definition, tools selection, and other architectural decisions, as well as team building, including handling the bureaucracy to hire the interns and the professionals from FLOSS communities and from the IT market.

The SPB portal project underwent two phases regarding the traceability of project management activities. The first one, between January 2014 and March 2015, is non-traceable, since only the universities managed the development activities. In April 2015, the project started using the SPB portal itself to manage the development process, inaugurating the second phase of the project. From then on, much of the management and communication activities were recorded and published in online channels and tools. During this period, the development leaders consolidated several FLOSS practices and agile values employed in the development process.

This study brings forth observations of researchers who worked directly in the project coordination, researchers who participated in the internal processes, and researchers who did not participate in the project. The first two groups mapped the practices adopted during the project and analyzed and explained the experience previously reported (Meirelles et al., 2017; Siqueira et al., 2018) by some of the authors of this paper. The others contributed their expertise in FLOSS and agile methods studies and helped the inside researchers define macro-decisions and interpret, with a more neutral perspective, the benefits and effects of these best practices on the project management context as preliminary presented by Wen et al. (2018).

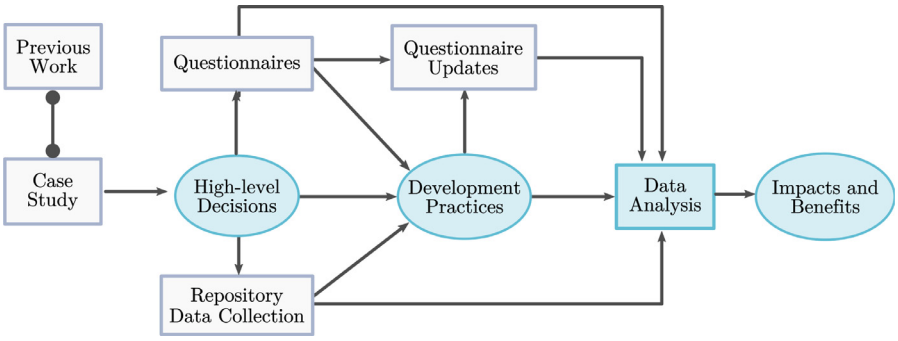

Fig. 1. Investigation workflow.

4.2.2. Survey

For us to obtain a good overview of how participants perceived multiple aspects of the project, we needed to survey them. However, as mentioned, the SPB project had a wide range of stakeholders. Accordingly, to guide the creation of the necessary questionnaires and interviews as well as the analysis of the collected data, we divided the project team into three groups: undergraduate interns, IT professionals (senior developers and designers), and MPOG analysts. Besides each member's point of view (obtained from interviews and questionnaires described here), we also had other sources of evidence, such as data from the code repository and from the activity management tool (as presented in Section 4.2.3).

We designed an online questionnaire[10] addressed to the undergraduate interns and another[11] addressed to the IT professionals, as well as semi-structured interviews (Edwards and Holland, 2013) with the two MPOG analysts who directly interacted with the development team and the project development process. To design the questionnaires and interview questions, we followed principles of survey research (Pfleeger and Kitchenham, 2001): searching the relevant literature, constructing the instrument (Kitchenham and Pfleeger, 2002a), and evaluating the instrument (Kitchenham and Pfleeger, 2002b). In particular, we used the lessons learned reported in our previous work (Meirelles et al., 2017) to reduce the overhead of survey construction (Kitchenham and Pfleeger, 2002a). Moreover, the authors of this study that participated directly in different roles of the project brought an improved comprehension of each interaction and aided in contacting the participants. Thereby, we conducted several brainstorming conversations with three ex-interns and two senior developers. This step, also recommended by the classical Ground Theory approach (Glaser and Strauss, 1999), provided a better understanding of the relevance of the project for the involved participants. All these principles and actions ultimately helped us shape the questionnaires and interview questions used as survey instruments.

Questionnaires

The questionnaire for the undergraduate interns dealt with topics related to (1) project organization, (2) development process, (3) communication and relationship with members, (4) acquired knowledge, and (5) experience with FLOSS projects. To cover these topics, we elaborated a total of 51 questions (45 multiple-choice questions and six open questions). The questionnaire for the IT professionals was comprised of 29 multiple-choice questions and 10 open questions, covering the following topics: (1) project organization; (2) project and stakeholders development processes; (3) communication and relationship with undergraduate interns; (4) the product developed; and (5) experience with FLOSS projects.

For both questionnaires, we designed multiple-choice questions for a more direct evaluation of the evolution of the methods used for activity management, of the organization of the project, and of the quality of the communication among all parties involved. We used open-ended questions to gather more nuanced perceptions.

The questionnaires were in Portuguese (the native language of the participants). The questionnaire guide translated into English is available in our repository[12]. We used the Google Forms service[13] to implement them and obtain response charts[14]. Before submitting the questionnaires to the participants, we performed a test-completion of both to estimate response time. We also conducted a trial with Computer Science Graduate Students at the University of São Paulo and IT professionals with similar profiles to the respondents. Thereby, we could confirm the readability and correct interpretation of the questions as well as fix spelling mistakes and replace ambiguous terms found by testers. We also redefined some response formats: For instance, we opted for the Likert scale over a numerical range in opinion closed-questions, because the Likert scale makes the respondent more comfortable to decide on a suitable answer.

We sent the questionnaires to 42 interns and 8 IT professionals. All interns worked as developers and received scholarships. We got a total of 37 (88%) intern responses, while all 8 IT professionals responded. On average, interns were 22 years old and professionals were 30 years old, wherein 3 of 37 (8%) and 1 of 8 (13%) respectively were women. 16 of 37 (about 43%) of the interns had the SPB project as their first contact with FLOSS. On average, the IT professionals had 11 years of experience, worked in at least 5 different companies, and participated in 4 to 80 distinct projects. Finally, 7 of 8 (86%) of them had some background with FLOSS before the SPB project.

Interviews

We conducted semi-structured interviews (Edwards and Holland, 2013) with the two MPOG analysts who directly interacted with the development team and the project development process. Semi-structured interviews are more spontaneous than structured interviews: They allow much more space for interviewees to answer on their terms. In a typical semi-structured interview, the interviewer follows an interview guide but is flexible to deviate from the predefined topical trajectory when he or she finds it appropriate and productive. He or she can probe answers, ensue a dialogue, and discuss a proposed theme better.

[10] https://gitlab.com/ccsl-usp/spb-jss-2018/raw/master/data-collection/questionnaires/interns-questions-1.md.

[11] https://gitlab.com/ccsl-usp/spb-jss-2018/raw/master/data-collection/questionnaires/senior-developers-questions.md.

[12] https://gitlab.com/ccsl-usp/spb-jss-2018/tree/master/data-collection/questionnaires.

[13] docs.google.com/forms.

[14] https://gitlab.com/ccsl-usp/spb-jss-2018/tree/master/data-collection/questionnaires/responses.

We initially defined an interview protocol with 28 questions in Portuguese, and shared it with respondents. The protocol translated into English is available in our repository.[15] The questions guide had four parts: (1) Professional profile;(2) Organization, communication, and development methodologies; (3) Satisfaction with the developed platform; and (4) Lessons learned. Before conducting the interviews, we reviewed the protocol with one project coordinator and four development team coaches who interacted directly with the MPOG analysts. Being a semi-structured interview, the interviewer conducted the interview placing each question according to the pace and meaning of each response. The interviewer also added additional questions during data collection to clarify or detail some interesting things mentioned by the interviewee.

Each interviewee was invited by email to indicate the date, time, and location most convenient for the interview. The interviews were conducted separately with each analyst via videoconference, using the Google Hangouts platform. Each meeting took an average of two hours and was carried out outside the working environment. Respondents agreed to have their voice and image recorded for this study. The interviewer was a member of the project team and had already met the interviewees at an earlier time. Both respondents were over 30 years old and had more than seven years working in the government. The analysts said that the SPB project represented their first experience of government-academia collaboration.

After two years of the project conclusion, we sent a second questionnaire[16] to the same group of undergraduate interns to ratify the results obtained in the previous survey and evaluate the impacts of the project experience on the professional life of these interns. This was a short-questionnaire with two closed questions, aimed at drawing a brief updated profile of the participants, and three open questions about the professional activities performed by them and the unfolding of the knowledge acquired in the project on these activities. We made the questionnaire available to receive responses for seven days, reaching 28 respondents of the 42 interns.

4.2.3. Repository data collection

Although most of the development team already had experience with agile methodologies and the use of tools such as Redmine[17] and GitLab in the development process, these resources were not part of the government's administrative culture. Consequently, in the first phase of the project (as explained in Section 4.2.1), these differences in managerial culture, coupled with the trust relationship still under construction between the two institutions, led to non-integrated management processes in the same project. The communication between government and academia was, generally, through private channels, such as professional e-mails, personal meetings, and telephone calls. Therefore, the quantitative data found for this period are not conclusive or have little expressiveness, and we do not examine them.

In contrast with this, the second phase of the project, when management activities shifted to the SPB Portal tools, was characterized by the automatic collection of a significant amount of meaningful data. This data source enabled us to reach a wealth of managerial information available on the project platform. We manually analyzed these data from the central project repository considering all the issues and commits from this second phase.

The project code was structured in the GitLab instance of the SPB platform. The development team defined a central repository, which concentrated on the customizations of the software. They also had local mirrors for the repositories of each integrated tool, where the team developed general improvements of each project for later submission of contributions. Within this structure, the dialogue between government and academia always took place around the central repository. The data collected from the repositories, which we discuss in Section 5.1.1, was divided into two categories:

Development data

- The number of commits per project made in the GitLab instance of the platform (Central, Noosfero, Colab, etc.).
- The number of different authors of commits registered in the repositories.
- The number of issues per project (Central, Noosfero, Colab, etc.).
- The number of different authors of issues with their respective identifier and author name.

Interaction data

- List of issues per project, ordered by ascending date.
- Date of the first issue opened by some MPOG staff (author).
- The number of comments on each issue, with names and emails of the participating authors.

4.2.4. Data analysis

Software engineering researchers have adopted Grounded Theory in recent years and regularly use it to analyze interviews and descriptive field-notes (Sbaraini et al., 2011). With a *coding* strategy from the Grounded Theory approach (Glaser and Strauss, 1999; Corbin and Strauss, 2014), we broke down and labeled the data into smaller components (Sbaraini et al., 2011). Based on this strategy, our analysis comprehends five steps:

(i) Data review: We organized and consolidated the data from the questionnaires and interviews (described in Section 4.2.2). During the process, we reviewed the records again, transcribing the full interviews. These transcripts preserve the interviewed speech, maintaining details such as colloquial expressions.

"A very positive meeting was about software qualification, talking to the team to get a level of detail of the feature they could understand, and all the questions were answered."

After that, we read the responses of the questionnaires open-questions. We reviewed the transcriptions and the form responses to fix typos, fill in small gaps in sentence construction, and map non-familiar terms. An example of non-familiar terms, according to the above transcription quote, is:

Software Qualification → *The feature developed to evaluate software available on the SPB platform.*

We also unified the meaning of some terms found:

Version Control: *GitLab, version management, versioning.*

Continuous Delivery: *setup management, Devops, automation, integration delivery, continuous delivery, automated tests.*

(ii) Practices and Benefits – Open coding: We identified expressions (codes) that could relate to the appropriate actions or perceptions described in the descriptive field-notes and the interviews. These initial codes are labels that represent the first set of mapped practices and benefits to be consolidated throughout all the analysis. This step is exemplified as follows – examining a piece of transcriptions related to the "government-academia interaction":

(a) "..robust interaction *via the Internet*"
(b) "..almost in *real-time*."

[15] https://gitlab.com/ccsl-usp/spb-jss-2018/raw/master/data-collection/interviews/mpog-interview.md.
[16] https://gitlab.com/ccsl-usp/spb-jss-2018/raw/master/data-collection/questionnaires/interns-questions-2.md.
[17] https://redmine.org.

(c) "..use of *mailing lists*, [..] *personal email*, [..] *Hangouts chat*."
(d) "..a lot of *discussion in the project itself* (with GitLab subsystem)."
(e) "..*video-conferences* to answer questions with the whole team distributed between 3–4 cities."
(f) "I had *no communication problem*."
(g) "..the *frequency of dialogues increased and evolved when the communication has migrated to the GitLab*."
(h) "..the interaction itself was *open and accessible*."
(i) "..the interaction was mostly *by email, GitLab, hangout, and even phone calls*."

Given the list above, we could extract the practices related to "government-academia interaction" from items (a), (c), (d), (e), (i), while the benefits derived from these practices are described by (b), (f), (g), (h).

We also examined the responses to the questionnaire open-questions related to "government-academia interaction" to map practices and derived benefits:

(a) "*Joint planning* and *seasonable meetings* were essential for *understanding MPOG's needs*."
(b) "*Interaction via SPB tools* helped *validate the system* as a development platform."

(iii) Decisions – from codes to categories: Based on the initial list of practices and benefits (the codes), we conducted a comparative analysis to identify similar expressions in the data, categorizing the labels to group codes of the same semantics. We can exemplify this process by taking the above example of open coding of transcription. Practices outlined in items (a), (c), (d), (i), and consequently the related benefits (f), (g), (h), fell under the same category related to "Government and Academia interact through the system under development". By completing this process, all practices and benefits reached some category (corresponding to one of the decisions discussed in Section 5.1).

(iv) Decisions refinement: In the previous step, we determined three categories. In this step, we did not change the number of mapped decisions. However, we renamed them to better describe each one, such as ***(category)***"*Government and Academia interact through the system under development*" to ***(decision)***"*Use of the system under development to develop the system itself*".

We also consolidated the group of practices and benefits related to each decision to support the found categories (summarized in Table 1, Section 5).

(v) From decisions to long-term benefits: Having identified the refined decisions, we surveyed the ex-interns (former undergraduate students), who joined the development team, in a second round. We examined how the identified decisions and practices reflected in their professional life after the project. Thereby, we obtained enough data to highlight the "Benefits for Teaching Software Engineering" as discussed in Section 5.2.2.

5. Results and discussion

As previously shown in Fig. 1, we extracted a set of high-level decisions presented as beneficial to the project from the lessons learned reported by academic coordination in the first study on the project (Meirelles et al., 2017). Based on the material just discussed, we analyzed the data obtained to understand the managerial practices, their consequences for the project, and to classify practices and benefits in a more consolidated description of the high-level decisions. This analysis resulted in an organized view of the core decisions made in the project, the practices that embodied these decisions throughout its evolution, and the perceived benefits brought by them.

Table 1 summarizes the results, organizing them in three columns. The first one presents the high-level decisions identified by the letter "D" plus the number of the decision (e.g., "D1"). The second column shows the managerial practices related to each decision. Its identifier includes the letter "P" plus the numbers of the related decision and the practice (e.g., "P1.1"). The last column presents the benefits derived from each decision and its associated practices. We labeled its identifier with the letter "B" plus the numbers of the related decision and the benefits (e.g., "B1.1"). We will use these identifiers to refer to the decisions, practices, and benefits in several parts of the discussions of the results in this section.

5.1. Decisions and benefits

In the case study, we identified three high-level project decisions that were relevant to reduce the communication gap between the teams and to increase project productivity:

D1 Use of the system under development to develop the system itself;

D2 Bring together government staff and development team;

D3 Organized development team into priority fronts, and for each one, hire at least one specialist from the IT market.

We now describe each of these decisions and their benefits according to the respondents, and present an analysis of corresponding repository data.

5.1.1. Decision (D1): use of the system under development to develop the system itself

The new SPB Portal combines functionalities adapted from existing collaborative software and others developed by the project team. As a result, the platform integrates different features such as social networking, mailing list, version control system, content management, and source code quality monitoring.

Because of these features, the development coordinators decided to use the platform under construction to develop the system itself. Gradually, in addition to development activities, the government and university teams moved the project management and communication between them to the portal environment.

Features like help pages, mailing list management, project-centered administration, Mezuro code quality analysis, and global search, offered the expected practical support for the development process. This use was also instrumental for the ongoing testing and improvement of the platform.

Collaborative Development Features supporting D1

The SPB Portal became a platform to stimulate (1) the openness of the source code, (2) an ongoing dialogue between users and the development team, and also (3) maintenance and evolution of the software, which provide more transparency in Government investments regarding software development. These qualities were offered mainly by use of the Collaborative Development Environment tools inherited from the GitLab integration.

Usually, CDEs demand a version control system, trackers, build tools, knowledge centers, and communication tools (Lanubile et al., 2010). The new SPB Portal also provides tools to encourage developers to keep the source code and its development activity within the platform. Any software project created in the SPB Portal has, by default, an associated Git repository with Wiki pages and issue tracking. The tools most often used during the development of the new SPB were:

1. Project repository: the teams used one organization with many repositories;
2. Wiki: each release had one Wiki page with the compilation of the strategical meeting notes;
3. Milestones: each milestone was used to register a user story (feature);
4. Issues: each sprint planning generated issues, which were associated to the related milestone (feature as user story)

Table 1
Empirical SPB management decisions and its benefits.

Decision	Practice Explanation	Benefits
D1: Use of the system under development to develop the system itself	• P1.1: The features and tools of the platform under development support the project management and communication activities.	• B1.1: Communicating with transparency and efficiency. • B1.2: Easy monitoring of activities. • B1.3: More interactions between developers and public servants. • B1.4: Confidence in the developed code. • B1.5: Organic documentation.
D2: Bring together government staff and development team	• P2.1: Government staff, academic coordinators, senior developers, and team coaches meet biweekly at the university lab for sprint planning and review. • P2.2: Direct contact, with no bureaucratic barriers, between government staff and the development team on the platform technical discussions. • P2.3: Involve government board of directors only in strategic planning of the project. • P2.4: Build a Continuous Delivery pipeline with stages involving both sides.	• B2.1: Reducing communication misunderstanding. • B2.2: Better meeting expectations of both sides. • B2.3: Improvement of the decision-making process. • B2.4: Overcoming the government bias regarding low productivity of collaborative projects with academia. • B2.5: Synchronizing the execution pace of activities. • B2.6: Shared responsibility using Continuous Delivery. • B2.7: Strengthening trust in the relationship with the government. • B2.8: Sharing a common understanding of the process from one side to the other.
D3: Organize the development team into priority fronts, and for each one, hire at least one specialist from the IT market	• P3.1: Coordinators separate the development team into priority work areas considering the main demands of the project. • P3.2: IT market professionals with recognized experience on each front were hired to work in person or remotely. • P3.3: Identify among the interns the leadership roles: a coach for each front, and a meta-coach of the entire development team. • P3.4: Each team develops its self-organization, being guided by one intern-coach and at least one senior developer.	• B3.1: Conciliating the development processes of each institution, taking better technical decisions. • B3.2: Improving the management and technical knowledge. • B3.3: Promoting team self-training with knowledge transfer. • B3.4: Providing opportunities for more advanced undergraduates to evaluate management issues and participate in business decisions. • B3.5: Self-organizing and gaining autonomy in the management of their tasks. • B3.6: Channeling professors' efforts to address high-level issues and to manage collaboration bureaucracies.

and registered on the corresponding Wiki page. Finally, each developer assigned the issue to him or herself.

The features provided by GitLab were particularly crucial in the management of the project. In short, the wiki feature was used for logging meetings, defining goals, planning sprints, and documenting deployment procedures and user guides. The issue tracker supported discussing requirements, monitoring features under development, requesting and recording changes, and validating the delivered functionalities. Finally, the mailing list was used for the collaborative construction of requirements, defining schedules, and scheduling meetings between institutions.

Thanks to the variety of features present in the platform, it was easily adapted to manage project planning, documentation, and development. Both university and government teams navigated and made use of these features daily. In general, the assimilation of the created functionalities happened organically, and the constant use of the development tools enabled the project members to test and validate their correct operation from the view of the developer, the user, and also the client.

Benefits brought by D1

Our surveys report the mailing list (100%) and issue tracker (62.5%) as the primary means of interaction between senior developers and interns. The development team and MPOG staff also interacted mostly via the mailing list (87.5%) and issue tracker (50%). According to one of the interviewees, this decision made the **communication more transparent and efficient** (B1.1). An MPOG analyst said that:

> *"Communicating well goes far beyond speed. It means enabling someone to tell everyone about everything that is happening in the project. We did not use [private] emails, we use more mailing list and avoid [private] emails. This usage helped us considerably. Everything was public and did not pollute our email box. So, when you wanted to know something, you could access the SPB list [on the web] and see everything".*

Migrating to the SPB platform also **eased monitoring of activities** (B1.2) and **increased interactions between developers and public servants** (B1.3). The data collected from the repository (Section 4.2.3) highlights the frequent use of the platform by both teams.

Significant usage by the development team was to be expected and, indeed, they made 3256 commits in the repository mentioned above. More importantly, there was strong participation from the government team as well: from April 2015 to June 2016 (the last 15 months of the project), 59 distinct authors, 8 of which were MPOG agents, opened 879 issues. These issues received a total of 4658 comments from 64 distinct users, 9 of them from MPOG. When we consider the issues with more interactions — those which had ten comments or more —, we notice that the government team also felt comfortable in using the tool to interact directly with the development team. In a set of 102 active issues, MPOG staff created 43 of them (this represents 42% of the most active issues).

For the MPOG analysts, interaction via repository improved communication:

> *"There was a big evolution, we increased our communication via GitLab".*

Migrating to the platform also led MPOG staff to **trust the developed code** (B1.4):

> *"Everything was validated. We tested the functionalities and developed the project on the SPB platform itself. Hence, the use of the system validated most of its features. From the moment we*

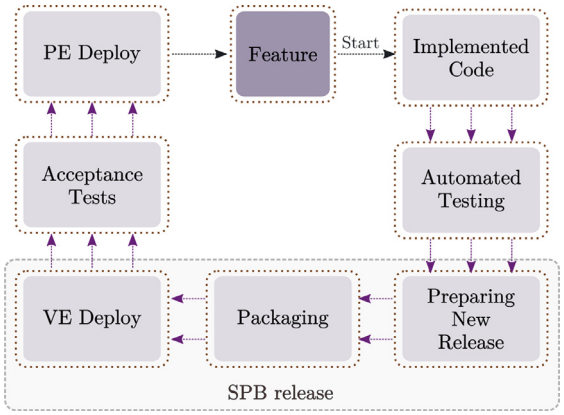

Fig. 2. The SPB Deployment Pipeline.

began to use it for development, this validation was constant. We felt confident in the code produced".

The decision mentioned above also collaborated to meet the government's demand for meticulous documentation of the software design and stages of development without bureaucratizing or modifying the development process. The usage of the platform for project team management conducted the **organic production of documentation and records** (B1.5), as mentioned in one of the MPOG responses:

"It was a great learning experience. There are many things documented in emails as well as in the portal itself. We can access the tools at any time and find out how we developed a solution. We can remember the positive points".

5.1.2. Decision (D2): bring together government staff and development team

As the project evolved, the MPOG analysts that we interviewed became responsible for negotiating with the teams the day-to-day development and product delivery decisions as well as for reporting to their superiors on inter-institutional meetings. During the first phase of the project, they did not participate in any direct interaction with university representatives. They stated that at that time there was significant communication noise in the internal dialogues with their superiors, as well as between them and the development team. It was in the project's second phase that those analysts became direct government representatives and started to visit the laboratory at the university biweekly. At this time, the development workflow had also changed due to the implementation of a Continuous Delivery (CD) pipeline. The development team used CD as a survival technique, that is, as a way to gain the government trust (Siqueira et al., 2018).

Continuous Delivery Pipeline supporting D2

The project's deployment followed a typical CD pipeline (Humble and Farley, 2010), adapted to the project's technical and organizational context and the use of FLOSS best practices. As depicted in Fig. 2, it began when a new feature is ready for deployment and ended when it reached production.

The implemented code was heavily tested. Each integrated system had its own test suite, and there was also a test suite for the platform as a whole. If any test suite failed, by either a test error or coverage reduction below a certain threshold, the process stopped. Only when all tests passed, the pipeline could proceed to prepare a new release.

SPB had its own Git repository.[18] An SPB portal release was an aggregate of all its integrated systems. That is, when one of them passed all of the SPB integration tests, the team manually created a corresponding tag on the SPB repository. At the end of this process, the DevOps team was ready to start packaging.

Packaging brings portability, simplifies deployment, and enables configuration and permission control. The approach used involved building separate packages for each system, in three fully automated steps: generating scripts for the specific environment, building the package, and uploading it to a package repository. When all ran successfully, the new packages would be ready and available for the deployment scripts. Before deploying to production, the system would first be deployed to a validation environment.

The Validation Environment (VE) was a replica of the Production Environment (PE) with anonymized data and access restricted to MPOG staff and the DevOps team. After a new SPB portal VE deployment, the development team used the environment to verify the integrity of the entire portal. MPOG staff also checked the new features, required changes, and bug fixes. If they identified a problem, they would notify developers via comments on the SPB portal issue tracker, prompting the team to fix it and restart the pipeline. Otherwise, the team could move forward to production deployment, using the same configuration management tool, scripts, and package versions as in the VE. After the deploy was completed, both VE and PE were identical, and any new features and bug fixes would become available to end users.

Benefits of D2

CD brings many advantages such as accelerated time to market, building the right product, productivity and efficiency improvements, stable releases, and better customer satisfaction (Chen, 2015; Savor et al., 2016). Fig. 3 shows the evolution of the number of releases during the project's lifetime both regarding its growth over time, grouped per semester (red bars) and depicts the number of project members (green line) and the reorganization of the team for the creation of a group specialized in DevOps (blue line).

Over 30 months, 84 versions were deployed. Without taking into account the first half of the project, where the team and the project requirements were still under construction, the project deployment averaged 15 versions per semester. After the creation of a DevOps team and the implementation of continuous delivery in June 2016, the number of deployments per semester grew to 27, representing an increase of 80% on the delivery rate. In summary, over 64% of the total of releases happened in these last 12 months of the project. It is also important to note that there is an intersection between DevOps members and undergraduate interns. Again, it is clear that, after an initial period dedicated to building and improving the CD pipeline, the creation of the DevOps team was crucial for the project to make more frequent releases.

Respondents also reported CD benefits. For 30 out of 37 (81%) of the interns and for 6 out of 8 (75%) of the IT professionals, deploying new versions of the SPB portal in production was a motivator during the project. On the government side, this approach helped to **overcome the government bias toward low productivity of collaborative projects with academia** (B2.4), as mentioned by them:

"Government staff has a bias that universities do not deliver products. However, in this project, we made many deliveries with high quality. Nowadays, I think if we had paid the same amount for a company, it would not have done the amount of features we did with the technical quality we have".

Additionally, constant deployments enabled both side teams to **share a common understanding of the process** (B2.8), because

[18] https://softwarepublico.gov.br/gitlab/softwarepublico.

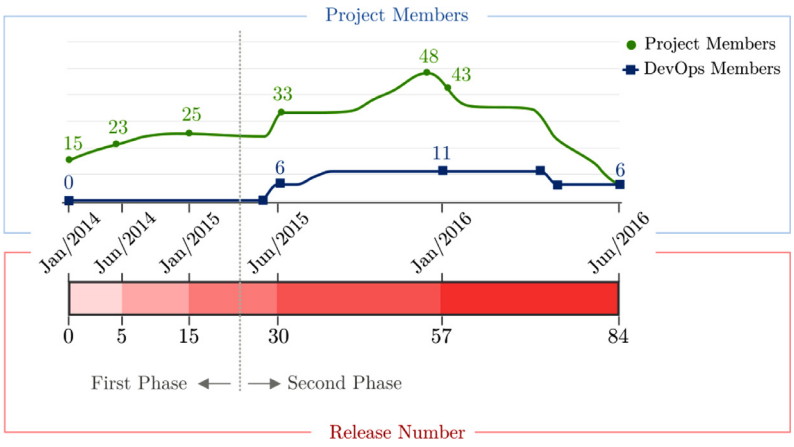

Fig. 3. The evolution of the SPB release in comparison to development team members distribution.

MPOG analysts became able to keep a high-level view of the project and to focus on planning future releases, as one of them mentioned:

> *"We had only the strategic vision of the project. When we needed to deal with technical issues, we had some difficulty planning the four-month releases. However, in the last stages of the project I realized that this was not a problem. The team was delivering and the results were available in production. The team was qualified, the code had quality, and the project was well executed. So in practice, our difficulty in interpreting the technical details did not impact the release planning".*

The presence of government representatives in the university laboratory was also responsible for reinforcing the government-academia synchronization. One of the analysts believed that:

> *"At this point, the communication started to change".*

The new dynamics **reduced communication misunderstandings** (B2.1) and unified both sides, as reported by another interviewee:

> *"It was very positive. We liked to go there and to interact with the team. I think it brought more unity, more integration into the project".*

27 out of 37 (73%) of the interns considered positive the direct participation of the MPOG staff, and 30 out of 37 (81%) of them believed the presence of government staff in sprint routines was relevant for the project development. For 28 out of 37 (76%) of the interns, writing the requirements together with the MPOG staff was very important to **meet expectations of both sides better** (B2.2). According to one of them, this closer interaction **improved the decision-making process** (B2.3):

> *"Joint planning and timely meetings were very important for understanding the needs of MPOG".*

Closer dialogue between government and academia generated empathy, as reported by one of the interviewees:

> *"Knowing people in person makes a big difference in the relationship because it causes empathy. You know who that person is. He's not merely a name".*

Consequently, this empathy helped to **synchronize the execution pace of activities** (B.5):

> *"Visiting the lab and meeting the developers encouraged us to validate resources faster and give faster feedback to the team. In return, they also quickly answered us any question".*

Some of the ex-undergraduate interns currently serve as academic professors or government employees. Due to the proximity between government and academia experienced during the project, such students reported encouraging the public administration to adopt agile methods, software sharing, and collaborative software development:

> *"The project experience developed my view about the value of FLOSS and software sharing between public administration departments. Hence, I have brought this concern inside the execution of my assignments and in discussions related to this theme with others civil servants."*

They also believe that these values, when implemented at the governmental level, enable resource savings and better project planning:

> *"The project enabled me to acquire a practical understanding of software development using agile methodologies and taught me the correct use of tools which bear this kind of development. Through this knowledge, we (MPOG) are spreading to other departments, and public servants the importance of agile methods and collaborative development for resource saving and software development planning inside the ministry."*

5.1.3. Decision (D3): Organized development team into priority fronts, and for each one, hire at least one specialist from the IT market

The SPB team was composed of a variety of professionals with different levels of experience and skill, where most of them were undergraduate software engineering students. Since students could not dedicate many hours per week to the project, they had the flexibility to negotiate their work schedule during the semester in order not to harm their classes and coursework. Their work routine in the project included programming and DevOps tasks.

The project required a vast experience and background that undergraduate students do not usually have. For this reason, a few senior developers have joined the project to help with the more difficult issues and to transfer knowledge to the students. Their main task was to provide solutions for complex problems, working as developers. As these professionals were very skillful and the project could not fund full-time work for all of them, they worked part-time on the project. Besides, they lived in either different Brazilian states or other countries, which led much of the communication to occur online.

Team Organization adopted with D3

Approximately 70% of the development team was composed of software engineering undergraduate students from UnB working physically in the same laboratory. The senior developers tried to

synchronize their work with the students' schedules, but each student had his schedule based on his classes, which complicated the adoption of pair programming. To cope with this scenario, a few basic rules were guiding the project organization:

1. Classes have high priority for undergraduate students;
2. Pairing whenever possible (locally or remotely);
3. During one morning or afternoon per week, *everyone* but the remote members should be together physically in the laboratory;
4. Every 2 to 3 months the senior developers would travel to work alongside the students for a few days.

With the aforementioned rules, the development team had four work areas divided in accordance to the main demands of the project: User Experience, DevOps, Integration of Systems, and Social Networking. One student of each team was the coach, responsible for reducing communication issues with other groups and helping the members to organize themselves in the best way for everyone (always respecting their work time). The coach also had the extra duty of registering the current tasks developed in the sprint. One important thing to notice is the mutability of the team and the coach. During the project many students switched teams to try different areas.

For each segment, at least one professional from the IT market was hired to raise the quality of the product assuming the position of senior developers. They have been selected based on their vast experience in FLOSS systems and their knowledge of the tools used in the project. Their expertise was essential to address hard decisions and complex problems. Thus, it was not the coach's role to deal with complicated technical decisions, which encouraged students to be coaches. Lastly, the senior developers worked directly with the students, and this was important to give them the opportunity to interact with a savvy professional in their areas and to keep the knowledge flowing in the project.

Finally, two other elements in the team organization were essential for the project harmony: the meta-coaches and the professors. The former were software engineers recently graduated that wanted to keep working on the project. The latter were professors that orchestrated all the interactions between all members of the project. Each meta-coach usually worked in one specific team and had the extra task of knowing the current status of all the others. Professors and the meta-coaches worked together to reduce communication issues among groups. Lastly, all the paperwork tasks, such as reporting on the project progress to the Brazilian Government, was handled by the professors.

Benefits of D3

The presence of senior developers in the project contributed to **conciliate the development processes of each institution and make better technical decisions** (B3.1), as quoted in one of the answers to the senior developer's questionnaire:

> *"I think my main contribution was to balance the relations between the MPOG staff and the university team".*

5 out of 8 (62.5%) of the IT professionals believe they have collaborated to conciliate the management and development process between the two institutions and another 5 of 8 (62.5%) helped MPOG staff express their requests more clearly. Government analysts were also more open to suggestions from these developers:

> *"They are upstream developers of the systems that integrate the platform. They conveyed trust, and therefore we trust the developed code".*

According to questionnaire responses, IT professionals mostly agreed with the project development process. For 5 out of 8 (62.5%), this process has close similarity to their previous experiences. In contrast, another 5 of 8 (62.5%) did not understand the MPOG's project management process and 4 out of 8 (50%) believed this process could affect the project productivity.

The senior developers were also responsible for **improving the management and technical knowledge** of the interns about practices from industry and open source projects (B3.2). 34 out of 37 (92%) of the interns believed that working with professionals was essential for learning and, for all of them, working with IT professionals was important during the project. 6 out of 8 (75%) of the IT professionals believed that "Working in pairs with a senior" and 5 out of 8 (62.5%) that "Participating in joint review tasks" were the activities with the involvement of them that most contributed to the evolution of the interns. 6 out of 8 (75%) believed that the knowledge shared by them with one intern was widespread among the others in the team. Government analysts also pointed out this knowledge sharing:

> *"On the university side, we noticed a significant improvement in the platform with the hiring of the systems' original developers. They had a guide on how to best develop each feature and were able to solve non-trivial problems quickly".*

Organizing the development team and hiring the IT professionals allowed each team to **self-organize and gain more autonomy in the management of their tasks** (B3.5). There was a development coach to lead each team, and at least one meta-coach supported all of them in their internal management activities. The coaches (most advanced interns) were important references in the development process. 33 out of 37 (89%) of the interns said that the presence of the coach was essential to the sprint's running, and for 7 out of 8 (88%) of the IT professionals the coaches were crucial for their interaction with the development team. MPOG analysts saw the coaches as facilitators for their activities and communication with the development team. They said:

> *"I interacted more with the project coordinator (professor) and team coaches", "Usually, we contact a coach to clarify some requirements or to understand some feature. The coaches were more available than senior developers and, sometimes, they would take our question to a senior developer".*

When we recently asked former undergraduate-interns about the experience acquired during the project, we observed that the team's organization and the presence of senior developers had brought maturity and positive impacts on their current professional activities. One of them stated that:

> *"The SPB project experience offered me a great maturity on professional FLOSS development".*

For another, the experience favored his career directly:

> *"The project experience helped me to develop myself technically and to understand some issues about project management of which I wouldn't have the opportunity in another moment. I also built a strong and valuable professional network with very qualified developers, and one of the senior developers introduced me to the Debian project (a Linux distribution). This gateway to the FLOSS environment was crucial for my current job in a recognized company of Open Source consultancy".*

Some respondents said that the experience with a partially remote team also contributed to their professional growth. In general, ex-interns have spontaneously reported that the contact of developers from different areas and professional profiles has enabled them to improve their technical skills and build a network of professional contacts:

> *"Besides the opportunity of technical contribution to large software projects, I learned a lot with senior developers as well as undergraduate colleagues. The project gave me the chance to work with*

various professional profiles (such as designers) and introduced me to the FLOSS world wherewith, and I am very grateful".

One ex-participant also reported that the organization of the team into priority fronts, and the possibility of migration between them, gave him a comprehensive view of how a quality product is developed. The experience of the project as a whole was the basis for some of them to be able to work directly on FLOSS projects - such as cooperating to Noosfero project, participating and mentoring in the Google Summer of Code Program, mentoring in the Outreachy Program,[19] and contributing to the Linux Kernel project.

5.2. Adopted practices and consequences

Providing a high-quality product and overcoming the differences in pace and interest of stakeholders were possible with the non-systematic employment of practices obtained from agile methodologies and FLOSS ecosystems. These practices emerged from the mentioned strategic decisions that guided the overall development of the project.

Our previous work (Wen et al., 2018) revealed a set of nine managerial practices that resulted in 14 benefits evidenced by replies to questionnaires and analysis of repository data. Table 1, from the previous section, presents an update of the summary of macro-decisions, practices, and benefits found in that work, as well as 5 new identified benefits. Based on the data set above, we analyzed all questionnaire's responses and performed a critical review of the effects of these practices on both the management of collaboration and the professional training of undergraduate students. After showcasing the major decisions and their benefits in the previous section, we now proceed to discuss the individual practices and how each one brought the benefits previously highlighted.

5.2.1. Day-to-day practices and benefits achieved

P1.1 *The features and tools of the platform under development support the project management and communication activities.* As soon as a first beta version of the new portal became available, the new SPB project, including both the development process and the project management activities, migrated to it. Discussions and technical decisions originally occurred in a dispersed way through difficult-to-monitor communication media such as telephone, e-mail, and face-to-face contact. The government and academia teams then started to keep their dialogues registered in the repository issues, wiki pages, and list of project discussions available on the platform.

- **B1.1** *Communicating with transparency and efficiency.* The use of a mailing list and the specification of requirements through issues on GitLab made the discussions and decisions available publicly and in real time.
- **B1.2** *Easy monitoring of activities.* Each code addition was versioned and directly related to the activity that originated it. Coordinators and developers could allocate contingency activities faster and monitor solutions adopted as well as all the code generated for it.
- **B1.3** *More interactions between developers and public servants.* Using online resources, made available in CMS and community format, broke physical and mental barriers of dialogue between the government team and the university team.
- **B1.4** *Confidence in the developed code.* Daily interactions with the software brought to the surface all bugs and accelerated the software improvement process.
- **B1.5** *Organic documentation.* Decisions, meeting reports, and changes were automatically recorded in the issue tracker, wiki, and mailing list.

P2.1 *Government staff, academic coordinators, senior developers, and team coaches meet biweekly at the university lab for sprint planning and review.* Initially, only the academic development team participated in the development follow-up routines, and leaders of academia moved to the government headquarters to meet the government team for determining the features to be delivered in the next four months. There was a gap in time and space between institutions that was softened by defining a more dynamic schedule, where the government team also attended meetings held at the university and participated more frequently in achieving the requirements.

- **B2.1** *Reducing communication misunderstanding.* In-person meetings generate empathy among team members and solve problems of interpretation due to errors in written expression and understanding thanks to speech tone.
- **B2.4** *Overcoming the government bias regarding low productivity of collaborative projects with academia.* Government staff realized the intensity of the agile development and shared the difficulties encountered during the construction of a feature.

P2.2 *Direct contact, with no bureaucratic barriers, between government staff and the development team on the platform technical discussions.* Stakeholders from both the government and the development team were encouraged to discuss issues and features directly among each other instead of going through the chain of command.

- **B2.2** *Better meeting of expectations on both sides.* Managing development through GitLab provide a greater approximation of all stakeholders to the development process, technical decisions, and deliveries of functionality.

P2.3 *Involve government board of directors only in strategic planning of the project.* Due to the directors' schedule, they could not keep up with the unfolding of technical decisions. To better explore the few meetings with the presence of the directors, discussions with them were restricted to strategic business issues, leaving the planning of the development process to the more frequent reunions with only the MPOG analysts.

- **B2.3** *Improvement of the decision-making process.* Technical details no longer interfered with strategic decisions, and political issues also did not affect technical choices. These restrictions generated less wear and tear among the teams and enriched the discussions of technical subjects among those better acquainted with them.

P2.4 *Build a Continuous Delivery pipeline with stages involving both sides.* The development and management processes became automated, providing guidance, clarity of responsibilities, and rhythm to all teams involved.

- **B2.5** *Synchronizing the execution pace of activities.* The CD pipeline performance depended on the synchronization between the academia and government teams, as each party had to be prepared to take action as soon as the other concluded a given task. The use of an explicit CD pipeline helped identify critical points of delay, and increased productivity.
- **B2.6** *Shared responsibility using Continuous Delivery.* According to the conventional MPOG process, the development team could not track what happened to the code after its delivery, since their employees were the only ones responsible for deployment. The implementation of CD made the development team feel equally responsible for what was getting into production and take ownership of the project (Shahin et al., 2016).

[19] https://outreachy.org.

B2.7 *Strengthening trust in the relationship with the government.* With CD, intermediate and candidate versions became available to be validated in the biweekly meetings, allowing the government staff to perform small validations over time. Constant monitoring of the development work brought greater assurance to the MPOG leaders and improved the interactions with the development team.

B2.8 *Sharing a common understanding of the process.* The steady pace and faster deployment provided by the pipeline brought a better overall view of the project for all participants.

P3.1 *Coordinators separate the development team into priority work areas considering the main demands of the project.* The project was divided into four work fronts, two to address general concerns of User eXperience and DevOps and two aimed at the development of the integrated applications that provided substantial functionalities to the platform: Noosfero and Colab.

B3.1 *Conciliating the development processes of each institution, taking better technical decisions.* With this division, both the government and academia teams began to describe better the requirements for identifying the responsible work front and correctly reference feedback, bugs, and necessary adjustments. Each work front has become a core of knowledge, developing among them technical skills according to the work scope. This improved the argumentative and explanatory capacity of the development team to technical issues raised by the government team.

P3.2 *IT market professionals with recognized experience on each front were hired to work in person or remotely.* With the division of the team, experienced professionals in the field of information architecture, design, front-end, infrastructure, as well as creators and maintainers of Noosfero, Colab, and other FLOSS projects joined to undergraduate interns in the different work fronts. Due to the high cost of moving this category of professionals, many worked remotely for the project.

B3.2 *Improving the management and technical knowledge.* The contracted professionals brought the team their technical abilities and experiences with industry, large-scale project and collaborative development. In addition to developing in the team the ability to organize and interact remotely, increasingly common in today's work environments.

B3.3 *Promoting team self-training with knowledge transfer.* Through peer programming, code review, and consultations, senior developers have transferred their knowledge to interns. Undergraduate interns could organically absorb a high load of technical expertise during the performance of their activities which would not be possible in the classroom context.

P3.3 *Identify among the interns the leadership roles: a coach for each front, and a meta-coach of the entire development team.* The academic coordinators invited the interns who had good maturity and knowledge of their work front to leadership and coaching roles. The one who acquired a good perception of the whole development process was then chosen to lead the dynamics between teams and the dialogues between the development team and the government team.

B3.4 *Providing opportunities for more advanced undergraduates to evaluate management issues and participate in business decisions.* When the intern took on the role of coach, he represented the team in the overall planning of the development of requirements and at times participated in direct meetings with government staff and board directors. These interactions enabled him to develop leadership and management skills.

P3.4 *Each team self-organizes, being guided by one intern-coach and at least one senior developer.* The intern-coach was responsible for planning and managing the activities and relied on senior developers' maturity and experience to schedule tasks, identify bottlenecks and overcome technical difficulties.

B3.5 *Self-organizing and gaining autonomy in the management of their tasks.* The closeness and openness in communication between the coach with the other interns summed with the maturity and market experience of the senior developers provided a balance in the development of the tasks of each work front and self-organization of the team according to the abilities of each one.

B3.6 *Channeling professors' efforts to address high-level issues and to manage collaboration bureaucracies.* Self-manageable work fronts allowed the academic coordinators (university professors) to be more available for definition of strategic goals with the government and management of the financial and contractual project issues, without compromising their teaching activities at the university.

The results presented in this paper corroborate the lessons learned in our previous work on studying the SPB project case (Meirelles et al., 2017). Evidence from the data collected, responses to questionnaires, and interviews reinforce what has been reported by the academic coordination of the project, adding the point of views of government and other roles involved on the academic side. In short, the government staff took time to understand how collaboration works and to realize that the project should not assume a client-executor relationship, but rather that both organizations were at the same hierarchical level in the work plan.

5.2.2. Benefits for teaching software engineering

After about two years of SPB project conclusion, in addition to contributions for the management of future partnerships between government and academia, this experience reverberated in the professional life of both senior developers and government agents, as well as the interns – former undergraduate students. The experience of a real project during the academic training allowed these newly graduated students to reach advanced positions in the universe of academia, private initiative, entrepreneurship, and FLOSS. The ex-interns revealed in their responses of the second round questionnaire that:

> *"The project helped me develop my masters and personal projects.", "It was the gateway to the job market and contributed significantly to my current job".*

71% of these ex-interns stated that the use of agile methods in a large-scale project with real clients was the primary professional contribution of the project since such practices are part of their current workflow.

> *"Everything that I learned during the SPB project I could employ on my current job, from best development practices to the use of Colab.", "The project experience was essential to start in the job market, especially the experience with agile methodologies and their* procedures".

68% of them have reported that DevOps abilities and versioning using git are knowledge from SPB project that improves their current work performance. For 36%, understanding of the collaborative development and the cycle of contributions to FLOSS projects are insights they apply in their professional and academic activities.

"Nowadays I am in the development area, and the DevOps knowledge acquired during the project helps me identify and solve problems faster.", "The knowledge of real projects, deliveries, scopes, and deadlines was essential to my professional performance".

In summary, because the SPB portal platform was not a "toy project", the heterogeneity of its project actively encouraged these ex-interns to work on their personal and professional maturity and to search for practical techniques of versioning, infrastructure, and DevOps.

"The experience with a multidisciplinary team reflects directly on my current job.", "The project experience was essential to define what I want to work with during my life and gave me the framework not to start my professional activities without a sense of real software development."), *"Now I can see the project as a substantial professional and personal growth experience. The integration between senior developers and undergraduate interns ripened the team in different moments, helping the team to uncover problems promptly, and search for reasonable alternatives or best ways to change. The adoption of agile practices also led us to a good dynamic of development".*

In addition to the delivered new version of the new SPB portal itself, this study also reveals that the adopted practices have benefited the people involved in the SPB rebuilding project. After we analyzed the *post-mortem* data of the SPB project, we noticed this project extrapolated FLOSS and agile values by impacting the lives of the people involved. Most of the former interns answered the second questionnaire: a total of 29 respondents. Five of them are working in large companies or Brazilian government agencies, three became entrepreneurs, and 11 are Computer Science or Information System researchers, which means 19 out of 29 respondents (65%). Additionally, several of the participants still actively contribute to large FLOSS communities, such as Debian, Fedora, Linux, Spark, Elixir, among others. Senior developers from the SPB rebuilding project have recruited some of the ex-interns they helped to train to become their co-workers. In summary, the project benefited the training of several students significantly.

6. Related work

Büyüközkan and Arsenyan (2012) divide the literature on collaborative product development in three categories: papers that describe typical collaborative product development dynamics and success factors, usually case studies; papers that deal with the process of partnership formation, such as choice of partner; and papers discussing technological and methodological tools to improve collaboration. Our work fits in the first and third categories, as it is a case study that describes the progressive development and adoption of new technologies and management techniques as the project evolved. Therefore, in this section, we prioritize the literature dealing both with the adoption of novel management approaches and with co-development involving either academia or government.

Dittrich et al. (2003) discussed the co-development of services, citizenship, and technology in the e-Government context. From four different cases, they mapped similar issues regarding e-government development. Since e-government changes the paradigm of several public services, the authors claim that it requires coordination by means of design and construction of supporting technical and organizational infrastructures. Our findings show that the government, especially the analysts, acquired all the expertise to design and provide the technical and organizational infrastructure to the users of the new SPB portal. They were actively involved in the collaborative development process and used the new platform tools for their management and communication activities during project development. From the analysts' responses, we observed that they recognized the potential of the new platform in practice through the development of the project itself. Hence, they became advanced users of the new SPB portal before any other users. Based on this experience, they also increased their understanding of collaborative software development.

Co-development is also related to knowledge transfer. Wan et al. (2010) conducted an empirical study on the influence of knowledge transfer in Software Process Improvement (SPI). They proposed a conceptual framework composed of five elements: "transfer of knowledge, sources of knowledge, recipients of knowledge, the relationship of transfer parties, and the environment of transfer". The framework investigated ten key factors such as "ambiguity, systematism, transfer willingness, the capacity of impartation, the capacity of absorption, incentive mechanism, culture, technical support, trust, and knowledge distance". Among these, the authors found that a trust relationship among the involved teams influenced the knowledge transfer and concluded that incentive mechanisms impacted positively in the knowledge transfer regarding SPI. Both findings are something we also observed. Our work reports that continued delivery and joint participation by both sides at various stages of the development cycle have built trust between government and academia. This confidence aroused the interest of the development team in understanding and better meeting the bureaucratic demands of the government. Also, the government analysts recognized the importance of absorbing agile values and FLOSS to the dynamism of the workflow. In the responses to the interviews and questionnaires, we also observed both teams realized that the transparency and learning of different project management techniques made the work more fluid and productive. Senior developers were not only motivated by payment but also by the opportunity to get new contributors to evolve the projects for which they were creators or maintainers. Besides, they found a way to show the relevance of their area of expertise to customer satisfaction and product acceptance by teaching students.

Foos et al. (2006) studied key-relationships that influence the transfer of knowledge between two partners. They collected both qualitative and quantitative data and identified that "trust, early involvement, and the due diligence" leverage the transfer of technology and tacit knowledge. The study also evidences that managers and project leaders comprehend poorly the implicit knowledge transfer, content, and process. The authors found there are different perceptions of knowledge transfer and a lack of methods to manage it. The findings indicate that managers did not succeed in absorbing knowledge for long-term product management. Managers work based on well-defined schedule and scope requirements of their projects, being "rewarded on execution, timing, and budgetary compliance". Foos et al. (2006) indicate that managers extract from their partner "minimally what is required to commercialize the product" because they are not responsible for the "next-generation product that they may not be working on". Unlike what is claimed in that paper, government analysts who managed the SPB project gained much knowledge from the academic and development team. They incorporated several practices in their day-to-day activities, involving other MPOG staff to collaborate in the SPB project. They also reported that the availability of ample documentation from the project is a valuable asset they can refer to at any time, suggesting a long-term change in management strategies and knowledge. In fact, an analyst mentioned that the experience extracted during the SPB project is useful to her new position as a project manager in another Brazilian government ministry.

Our paper focuses on two particular partners: academia and government. Some works discuss how academia can collaborate and transfer knowledge to the industry in the management of software projects. Chookittikul et al. (2011) evaluated the increas-

ing use of agile techniques in software development companies in Thailand. The authors suggested that universities should create curricula that develop their undergraduate students' practical skills required by industry (mainly agile practices) to promote growth in local software businesses. Our findings from the second questionnaire sent to former undergraduates confirm the importance of including the development of practical skills in undergraduate curricula. The SPB project was a successful case of teaching and training IT professionals. The respondents reported that FLOSS and Agile methodologies skills developed during the project made it easier for them to get a job and also perform well their work activities.

Sandberg and Crnkovic (2017) report the use of Scrum in an industry-academia research consortium (involving ten industry partners and five universities in Sweden). Through a case study, they demonstrate that being able to bring together the meaningful activities of the stakeholders is essential to the success of collaborative research between industry and academia. Similarly to theirs, our work also shows how the adoption of agile values supports the collaboration of institutions with different goals. Sandberg and Crnkovic (2017) found nine practical ways to help industry-academia collaborations, including organizing activities in sprints, monthly face-to-face stakeholder meetings, using Scrum routines, joint development planning, and delivery of intermediate versions. Their results have a close connection with four practices reported in our paper that describe Decision 2 (D2) made in the SPB project.

Mergel (2016) conducted a study including interviews with managers of the central digital transformation team from the U.S. government. The author investigated the existing initiatives for agile management implementation and proposed a research framework to guide future investigations regarding collaborative and agile innovation management approaches in government. This framework includes research questions such as: "How can government incentivize open sharing of source code instead of reinventing the wheel with every request for proposals, signed contract or grant?". The SPB Portal itself, the target of our study, is a case study that answers this question: the portal's primary purpose is providing an environment to share software developed for or by any of the many Brazilian government agencies, reducing the costs of software and services acquisition. The development of the platform also addressed this point: the platform is a system-of-systems that has adapted and integrated five existing FLOSS projects, whose development was supported by the management decisions and practices listed in this study.

Discussions on how to introduce new management methods into an organization are present in the literature (Qumer and Henderson-Sellers, 2008; Misra et al., 2009; Mishra and Mishra, 2011). However, Nuottila et al. (2016) highlights that only a few works investigated agile adoption in public organizations. Based on a case study from the Finnish government, they conducted research "to identify and categorize the challenges that may hinder efficient adoption and use of agile methods in public IT projects". The study presents seven challenges: Documentation (C1); Education, experience, and commitment (C2); Stakeholder communication and involvement(C3); Roles in the agile set-up (C4); Location of the agile teams (C5); Legislation (C6); Complexity of software architecture and system integration (C7). The decisions discussed in our paper are directly related to the challenges C1, C2, C3 and C7. Our Decision 1 (D1) provided benefits such as organic documentation (B1.5), communicating with transparency and efficiency (B1.1), and more interactions between developers and public servants (B1.3). The benefit B1.5 is related to challenge C1 while B1.1 and B1.3 concern C3. Benefits from Decision 2 (D2) such as reducing communication misunderstanding (B2.1), better meeting expectations of both sides (B2.2), synchronizing the execution pace of activities (B2.5), strengthening trust in the relationship with the government (B2.7), and sharing a common understanding of the process from one side to the other (B2.8) also are related to challenge C3. Our Decision 2 (D2), in specific the practice P2.4 (build a Continuous Delivery pipeline with stages involving both sides), also provided benefit B2.6 (Shared responsibility using Continuous Delivery), associated with challenge C7. Lastly, Decision 3 (D3) brought benefits such as improving management and technical knowledge (B3.2), promoting team self-training with knowledge transfer (B3.3), providing opportunities for more advanced undergraduates to evaluate management issues and participate in business decisions (B3.4), and self-organizing and gaining autonomy in the management of their tasks (B3.5) which correlates to challenge C2.

Finally, Nerur et al. (2005) and Strode et al. (2009) (discussed in Section 2) argue that the adoption of agile development techniques does indeed produce changes in an organization's culture. Conversely, our study does not discuss significant changes in the involved Brazilian government agency, since the strategy reported here had the premise that both organizations (academia and government) should collaborate with minimum impact on their internal macro-management processes and culture. As seen in this section, such form of collaboration yields positive outcomes mostly similar to other endeavors.

7. Conclusion

We reported on our investigation on the rebuilding of the Brazilian government platform for software sharing, the SPB portal. For restructuring the architecture and providing new functionality to the system, the federal government teamed up with universities in a 30-month collaboration project. The partnership developed a unique platform, described by project members interviewed as an innovative product, with no other similar reference within the Brazilian government. Due to the adoption of agile and FLOSS practices and collaborative development technologies, the software delivered can be easily updated and replicated for other government sectors, and the management structure may serve as an example for future government-academia collaborations.

In addition to technical aspects of the platform, the collaboration studied in this work had a significant managerial complexity. By involving large-scale organizations that have significant differences in their development methods and customary workflows, the development process needed to be well-adapted to mitigate mismatch and conflicts of interest, without interposing their internal managerial procedures and organizational routines. This conciliation is crucial in a case of government-academia collaboration since the weak and unadaptable management could lead the project to fail, resulting in the waste of tax-payer resources.

The case study enabled us to systematize management best practices that mitigate the interinstitutional conflicts and led the partnership to succeed. As a result, we identified high-level decisions, empirically taken by the development leaders, which oriented the whole team to adopt practices from agile methods and FLOSS ecosystems. From this, we asked members on both sides of the project how these decisions had influenced the progress of activities. The answers obtained, together with data collected from the development repositories, enabled us to map the best practices and the benefits they generated.

Regarding our first research question *"How to introduce FLOSS and agile best practices into government-academia collaboration projects?"*, we examined the SPB project and identified three high-level decisions taken by the academic coordinators that drove them to intuitively adopt nine FLOSS and agile best practices in the development process:

D1 Use of the system under development to develop the system itself.
D2 Bring together government staff and development team.
D3 Organize development teams into priority fronts, and for each one, hire at least one specialist from the IT market.

The interview responses and data collected from the development repositories enabled us to understand how FLOSS and agile practices contributed to project management. We recently sent ex-interns a set of complementary questions to update our analysis of the adopted management method benefits and the project experience impacts on their professional lives. Based on these data, we answered the second research question *"What FLOSS and agile practices favor effective team management in government-academia collaborative projects?"*, by enumerating at least 19 benefits obtained from the use of the nine best practices:

P1.1 The features and tools of the platform under development support the project management and communication activities.
P2.1 Government staff, academic coordinators, senior developers, and team coaches meet biweekly at the university lab for sprint planning and review.
P2.2 Direct contact, with no bureaucratic barriers, between government staff and the development team on the platform technical discussions.
P2.3 Involve government board of directors only in strategic planning of the project.
P2.4 Build a Continuous Delivery pipeline with stages involving both sides.
P3.1 Coordinators separate the development team into priority work areas considering the main demands of the project.
P3.2 IT market professionals with recognized experience on each front were hired to work in person or remotely.
P3.3 Identify among the interns the leadership roles: a coach for each front, and a meta-coach of the entire development team.
P3.4 Each team develops its self-organization, being guided by one intern-coach and at least one senior developer.

Moreover, based on the responses from the second questionnaire sent to the ex-interns, we were also able to evaluate the additional benefits in terms of training in software engineering via an experience on a real software development project. Interns were exposed to the complexities of large-scale institutions, advanced high-tech challenges, as well as the dynamics of FLOSS projects. About two years after the end of the project, the ex-interns who participated in the new SPB portal development reached significant positions in their academic or professional careers; several of them also have significant contributions to large FLOSS projects. Our study showed that the adopted practices impacted the people involved in the project notably.

This research has a few limitations. First, we point out the lack of formal communication records and low traceability of the management data referring to the first phase of the project. Second, we consider a drawback the hiatus between the completion of the project and the execution of interviews and questionnaires, since we relied on the memory of the interviewees to recollect events. Nevertheless, this hiatus also allowed team members more time to reflect on their participation in the project and let them gain more professional experience enabling them to better evaluate the period in which they participated in the project. Third, the current situation of the respondents, such as their current working mindset, may also alter their perception of the topics addressed in the questionnaire and, consequently, their responses. Finally, the decisions, practices, and benefits discussed should be evaluated and used in context with a more substantial plurality and diversity of government stakeholders.

We believe that our findings can be used as effective guidelines for government-academia collaborative software development projects. In fact, UnB and USP are currently adopting some of these practices to develop government-academia software projects in the fields of Healthcare and Culture in collaboration with federal and state governments. As future work, we believe it will be valuable to compare different approaches for managing government-academia collaborations and map other useful managerial methods, evaluating the short- and long-term organizational impacts on the institutions involved.

Acknowledgments

This research is part of the INCT of the Future Internet for Smart Cities funded by the Brazilian National Council for Scientific and Technological Development (CNPq) proc. 465446/2014-0, the Coordination for the Improvement of Higher Education Personnel – Brasil (CAPES) – Finance Code 001, and the São Paulo Research Foundation (FAPESP) proc. 14/50937-1 and proc. 15/24485-9. Some authors were supported by fellowships from CNPq and CAPES, Brazil. The authors thank especially all project ex-interns and IT professionals as well as the Brazilian government IT professionals for answering the questionnaires, participating in the interviews, and being available to improve our understanding of the SPB rebuilding project. Their contributions during and after the project were essential for developing this research.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.jss.2020.110548.

References

Adams, P., Capiluppi, A., 2009. Bridging the gap between agile and free software approaches: the impact of sprinting. Int. J. Open Source Softw.Process. (IJOSSP) 1, 58–71. doi:10.4018/jossp.2009010104.
Anthopoulos, L., Reddick, C.G., Giannakidou, I., Mavridis, N., 2016. Why e-government projects fail? an analysis of the healthcare. gov website. Gov. Inf. Q. 33, 161–173.
Balter, B.J., 2011. Toward a more agile government: the case for rebooting federal it procurement. Public Contract Law J. 41 (1), 149–171.
Beck, K., Beedle, M., Bennekum, A., et al., 2010. Manifesto for agile software development. Agile Alliance (2001). Retrieved June 14.
Booch, G., Brown, A.W., 2003. Collaborative development environments. Adv. Comput. 59, 1–27.
Büyüközkan, G., Arsenyan, J., 2012. Collaborative product development: a literature overview. Prod. Plann. Control 23 (1), 47–66. doi:10.1080/09537287.2010.543169.
Capiluppi, A., Lago, P., Morisio, M., 2003. Characteristics of open source projects. In: Software Maintenance and Reengineering, 2003. Proceedings. Seventh European Conference on. IEEE, pp. 317–327.
Chandra Misra, D., 2006. Defining e-government: a Citizen-Centric Criteria-Based Approach.
Charmaz, K., 2008. Chapter 7: Grounded Theory as an Emergent Method. Handbook of Emergent Methods. The Guilford Press.
Chen, L., 2015. Continuous Delivery: Huge Benefits, but Challenges too 32.
Chesbrough, H., Schwartz, K., 2007. Innovating business models with co-development partnerships. J. Res.-Technol. Manage. 50, 55–59.
Chookittikul, W., Kourik, J.L., Maher, P.E., 2011. Reducing the gap between academia and industry: the case for agile methods in thailand. In: 2011 Eighth International Conference on Information Technology: New Generations, pp. 239–244. doi:10.1109/ITNG.2011.49.
Corbin, J., Strauss, A., 2014. Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory, 4th SAGE Publications, Inc.
Corbucci, H., Goldman, A., 2010. Open source and agile methods: two worlds closer than it seems. In: Sillitti, A., Martin, A., Wang, X., Whitworth, E. (Eds.), Agile Processes in Software Engineering and Extreme Programming. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 383–384.
DeKoenigsberg, G., 2008. How successful open source projects work, and how and why to introduce students to the open source world. In: Saiedian, H., Williams, L.A. (Eds.), CSEET. IEEE Computer Society, pp. 274–276.
Deshpande, A., Riehle, D., 2008. Continuous integration in open source software development. In: Russo, B., Damiani, E., Hissam, S., Lundell, B., Succi, G. (Eds.), IFIP International Conference on Open Source Systems: Open Source Development, Communities and Quality. Springer US, Boston, MA, pp. 273–280.

Dittrich, Y., Ekelin, A., Elovaara, P., Eriksen, S., Hansson, C., 2003. Making e-government happen everyday co-development of services, citizenship and technology. In: 36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the, p. 12. doi:10.1109/HICSS.2003.1174328.
Ducheneaut, N., 2005. Socialization in an open source software community: a socio-technical analysis. Comput. Supported Cooperat. Work 14 (4), 323–368.
Düring, B., 2006. Sprint driven development: Agile methodologies in a distributed open source project (pypy). In: Abrahamsson, P., Marchesi, M., Succi, G. (Eds.), Extreme Programming and Agile Processes in Software Engineering. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 191–195.
Edwards, R., Holland, J., 2013. What is Qualitative Interviewing?. A&C Black.
Fagerholm, F., Guinea, A.S., Münch, J., Borenstein, J., 2014. The role of mentoring and project characteristics for onboarding in open source software projects. In: Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. ACM, New York, NY, USA, pp. 55:1–55:10.
Fagerholm, F., Johnson, P., Guinea, A.S., Borenstein, J., Münch, J., 2013. Onboarding in open source software projects: Apreliminary analysis. CoRR. arXiv: 1311.1334.
Fitzgerald, B., Stol, K.-J., 2017. Continuous software engineering: a roadmap and agenda. J. Syst. Softw. 123, 176–189. doi:10.1016/j.jss.2015.06.063.
Foos, T., Schum, G., Rothenberg, S., 2006. Tacit knowledge transfer and the knowledge disconnect. J. Knowl. Manage. 10, 6–18. doi:10.1108/13673270610650067.
Fraser, S., Ågerfalk, P.J., Eckstein, J., Korson, T., Rainsberger, J.B., 2006. Open source software in an agile world. In: Abrahamsson, P., Marchesi, M., Succi, G. (Eds.), Extreme Programming and Agile Processes in Software Engineering. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 217–220.
Freitas, C., Meffe, C., 2008. FLOSS in an Open World: best practices from Brazil. Roadmap white paper. 2020 FLOSS Roadmap. Paris, France
Gary, K., Enquobahrie, A., Ibanez, L., Cheng, P., Yaniv, Z., Cleary, K., Kokoori, S., Muffih, B., Heidenreich, J., 2011. Agile methods for open source safety-critical software. J. Softw. 41 (9), 945–962. doi:10.1002/spe.1075.
Gerring, J., 2006. Case Study Research: Principles and Practices. Cambridge University Press.
Glaser, B., Strauss, A., 1999. The Discovery of Grounded Theory: Strategies for Qualitative Research. Aldine Transaction.
Goldfinch, S., 2007. Pessimism, computer failure, and information systems development in the public sector. Public Adm. Rev. 67, 917–929.
Goth, G., 2007. Sprinting toward open source development. IEEE Softw. 24 (1), 88–91. doi:10.1109/MS.2007.28.
Harzl, A., 2017. Can foss projects benefit from integrating kanban: a case study. J. Internet Serv. Appl. 8 (1), 7.
Hoda, R., Noble, J., 2017. Becoming agile: A grounded theory of agile transitions in practice. In: 2017 IEEE/ACM 39th International Conference on Software Engineering, pp. 141–151.
Humble, J., Farley, D., 2010. Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation, 1st Addison-Wesley Professional.
Javdani Gandomani, T., Zulzalil, H., abdul ghani, A., Md Sultan, A.B., 2013. A systematic literature review on relationship between agile methods and open source software development methodology. Int. Rev. Comput. Softw. 7.
Kitchenham, B., Pfleeger, S., 2002a. Principles of survey research: part 3: constructing a survey instrument. ACM SIGSOFT Softw. Eng. Notes 27, 20–24.
Kitchenham, B., Pfleeger, S., 2002b. Principles of survey research part 4: questionnaire evaluation. ACM Sigsoft Softw. Eng. Notes 27. doi:10.1145/638574.638580.
Kon, F., Meirelles, P., Lago, N., Terceiro, A., Chavez, C., Mendonça, M., 2011. Free and open source software development and research: opportunities for software engineering. In: SBES. IEEE Computer Society, pp. 82–91.
Kourtesis, D., Bratanis, K., Bibikas, D., Paraskakis, I., 2012. Software co-development in the era of cloud application platforms and ecosystems: the case of cast. In: Camarinha-Matos, L.M., Xu, L., Afsarmanesh, H. (Eds.), Collaborative Networks in the Internet of Services. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 196–204.
Lanubile, F., Ebert, C., Prikladnicki, R., Vizcaíno, A., 2010. Collaboration tools for global software engineering. IEEE Softw. 27 (2), 52–55.
Lavazza, L., Morasca, S., Taibi, D., Tosi, D., 2010. Applying scrum in an oss development process: an empirical evaluation. In: Sillitti, A., Martin, A., Wang, X., Whitworth, E. (Eds.), Agile Processes in Software Engineering and Extreme Programming. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 147–159.
Littler, D., Leverick, F., Bruce, M., 1995. Factors affecting the process of collaborative product development: a study of UK manufacturers of information and communications technology products. J. Prod. Innovation Manage. 12 (1), 16–32. doi:10.1111/1540-5885.1210016.
Magdaleno, A.M., Werner, C.M.L., de Araujo, R.M., 2012. Reconciling software development models: Aquasi-systematic review. Journal of Systems and Software 85 (2), 351–369. Special issue with selected papers from the 23rd Brazilian Symposium on Software Engineering doi: 10.1016/j.jss.2011.08.028.
Margetts, H., Dunleavy, P., 2013. The second wave of digital-era governance: a quasi-paradigm for government on the web. Philos. Trans. R. Soc. A 371 (1987), 20120382. doi:10.1098/rsta.2012.0382.
Meirelles, P., Wen, M., Terceiro, A., Siqueira, R., Kanashiro, L., Neri, H., 2017. Brazilian public software portal: an integrated platform for collaborative development. In: Proceedings of the 13th International Symposium on Open Collaboration. ACM, pp. 16:1–16:10.
Melo, C., Santos, V., Katayama, E., Corbucci, H., Prikladnicki, R., Goldman, A., Kon, F., 2013. The evolution of agile software development in brazil. J. Braz. Comput. Soc. 19 (4), 523–552. doi:10.1007/s13173-013-0114-x.
Mergel, I., 2016. Agile innovation management in government: A research agenda. Government Information Quarterly 33 (3), 516–523. Open and Smart Governments: Strategies, Tools, and Experiences doi: 10.1016/j.giq.2016.07.004.
Mishra, D., Mishra, A., 2011. Complex software project development: agile methods adoption. J. Softw. Mainten. Evol. 23 (8), 549–564. doi:10.1002/smr.528.
Misra, S.C., Kumar, V., Kumar, U., 2009. Identifying some important success factors in adopting agile software development practices. Journal of Systems and Software 82 (11), 1869–1890. SI: TAIC PART 2007 and MUTATION 2007 doi: 10.1016/j.jss.2009.05.052.
Müller, M., 2018. Agile challenges and chances for open source: Lessons learned from managing a floss project. In: 2018 IEEE Conference on Open Systems (ICOS), pp. 1–6. doi:10.1109/ICOS.2018.8632819.
Nerur, S., Mahapatra, R., Mangalaraj, G., 2005. Challenges of migrating to agile methodologies. Commun. ACM 48 (5), 72–78.
Nielsen, C.B., Larsen, P.G., Fitzgerald, J., Woodcock, J., Peleska, J., 2015. Systems of systems engineering: basic concepts, model-based techniques, and research directions. ACM Comput. Surv. 48 (2), 18:1–18:41.
Nuottila, J., Aaltonen, K., Kujala, J., 2016. Challenges of adopting agile methods in a public organization. Int. J. Inf. Syst.Project Manage. 4, 65–85. doi:10.12821/ijispm040304.
Okoli, C., Carillo, K., 2012. The best of adaptive and predictive methodologies: open source software development, a balance between agility and discipline. Int. J. Inf. Technol. Manage. 11 (1–2), 153–166.
Pfleeger, S., Kitchenham, B., 2001. Principles of survey research: part 1: turning lemons into lemonade. ACM SIGSOFT Softw. Eng. Notes 26, 16–18. doi:10.1145/505532.505535.
Porruvecchio, G., Concas, G., Palmas, D., Quaresima, R., 2007. An agile approach for integration of an open source health information system. In: Concas, G., Damiani, E., Scotto, M., Succi, G. (Eds.), Agile Processes in Software Engineering and Extreme Programming. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 213–218.
Qumer, A., Henderson-Sellers, B., 2008. A framework to support the evaluation, adoption and improvement of agile methods in practice. J. Syst. Softw. 81 (11), 1899–1919. doi:10.1016/j.jss.2007.12.806.
Rahman, S.S.M., Mollah, S., Anirban, S., Rahman, M., Rahman, M., Hassan, M.M., Sharif, M.H., 2018. Oscrum: a modified scrum for open source software development. Int. J. Simul. Syst. Sci. Technol. 19, 20.1–20.7. doi:10.5013/IJSSST.a.19.03.20.
Raymond, E., 1999. The cathedral and the bazaar. Philos. Technol. 12 (3), 23.
Sandberg, A.B., Crnkovic, I., 2017. Meeting industry: academia research collaboration challenges with agile methodologies. In: Proceedings of the 39th International Conference on Software Engineering: Software Engineering in Practice Track. IEEE Press, pp. 73–82.
Santos, R., Silva, F., Magalhaes, C., Monteiro, C., 2016. Building a theory of job rotation in software engineering from an instrumental case study. In: 2016 IEEE/ACM 38th International Conference on Software Engineering, pp. 971–981.
Savor, T., Douglas, M., Gentili, M., Williams, L., Beck, K., Stumm, M., 2016. Continuous deployment at facebook and oanda. In: Proceedings of the 38th International Conference on Software Engineering Companion.
Sbaraini, A., Carter, S.M., Evans, R.W., Blinkhorn, A., 2011. How to do a grounded theory study: a worked example of a study of dental practices. BMC Med. Res. Methodol. 11 (128), 1–20.
Scholl, H.J., 2003. E-government: a special case of ict-enabled business process change. In: 36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the, p. 12.
Shahin, M., Babar, M.A., Zhu, L., 2016. The intersection of continuous deployment and architecting process: practitioners' perspectives. In: Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, pp. 44:1–44:10.
Siqueira, R., Camarinha, D., Wen, M., Meirelles, P., Kon, F., 2018. Continuous delivery: Building trust in a large-scale, complex government organization. IEEE Software PP (99). 1–1
Steghöfer, J.-P., Knauss, E., Alégroth, E., Hammouda, I., Burden, H., Ericsson, M., 2016. Teaching agile: addressing the conflict between project delivery and application of agile methods. In: Proceedings of the 38th International Conference on Software Engineering Companion. ACM, New York, NY, USA, pp. 303–312.
Stol, K.-J., Ralph, P., Fitzgerald, B., 2016. Grounded theory in software engineering research: acritical review and guidelines. In: 2016 IEEE/ACM 38th International Conference on Software Engineering, pp. 120–131.
Strode, D.E., Huff, S.L., Tretiakov, A., 2009. The impact of organizational culture on agile method use. In: System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on. IEEE, pp. 1–9.
Tsirakidis, P., Kobler, F., Krcmar, H., 2009. Identification of success and failure factors of two agile software development teams in an open source organization. In: 2009 Fourth IEEE International Conference on Global Software Engineering, pp. 295–296. doi:10.1109/ICGSE.2009.42.
Wan, J., Liu, Q., Li, D., Xu, H., 2010. Research on knowledge transfer influencing factors in software process improvement. J. Softw. Eng. Appl.(JSEA) 3, 134–140. doi:10.4236/jsea.2010.32017.
Warsta, J., Abrahamsson, P., 2003. Is open source software development essentially an agile method. In: Proceedings of the 3rd Workshop on Open Source Software Engineering, pp. 143–147.
Waterman, M., Noble, J., Allan, G., 2015. How much up-front?: a grounded theory of agile architecture. In: 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, pp. 347–357.
Wen, M., Meirelles, P., Siqueira, R., Kon, F., 2018. Floss project management in government-academia collaboration. In: Stamelos, I., Gonzalez-Barahoña, J.M., Var-

lamis, I., Anagnostopoulos, D. (Eds.), Open Source Systems: Enterprise Software and Solutions. Springer International Publishing, Cham, pp. 15–25.
Yin, R., 2009. Case Study Research: Design and Methods. SAGE Publications.

Melissa Wen is currently a Computer Science MSc student at the Institute of Mathematics and Statistics of the University of São Paulo (IME-USP). She holds a BSc in Computer Science from Federal University of Bahia (UFBA), where she has also served as a temporary professor. She worked on the SPB project as a senior developer and was core developer of Noosfero social networking platform at Colivre. She also contributed to the GNOME project in the context of Outreach Program For Women. Her areas of interest include Software Engineering and Free Software Development.

Rodrigo Siqueira is Software engineer from University of Brasilia (UnB) and received his M.Sc. in Computer Science from the University of São Paulo (IME-USP). His research interests gravitate around Operating System, Software Engineering, Computer Architecture, and FLOSS. He worked for two years in the SPB project as a coach and developer. Currently, he actively contributes with Linux Kernel at the Direct Rendering Manager and also helping newcomers to become part of the kernel community.

Nelson Lago received his M.Sc. in Computer Science from the University of São Paulo (USP). He is currently the technical manager for the FLOSS Competence Center at the Institute of Mathematics and Statistics of the University of São Paulo (IME-USP). His research interests include FLOSS, Open Data, Privacy in the Digital Domain, and Computer Music.

Diego Camarinha Software developer with a MSc degree in Computer Science at the Institute of Mathematics and Statistics of the University of São Paulo (IME-USP). He worked for two years on the SPB project as a senior developer. His research interests include Free Software development, Agile Software development, Computer Networks, and Source Code Metrics.

Antonio Terceiro received his M.Sc. in Computer Science from the Federal University of Rio Grande do Sul (UFRGS) and his Ph.D. in Computer Science from the Federal University of Bahia (UFBA). Antonio is currently a Software Engineer at Linaro Limited, working on platforms for Software Quality Assurance, and is also a Free Software developer and member of the Debian Project. His interests include Free Software, FLOSS development, Software Engineering and Software Quality Assurance.

Fabio Kon is a Full Professor of Computer Science at the University of São Paulo (IME-USP) and Visiting Professor at the MIT Senseable City Lab. He was a cofounder of the Free/Libre/Open Source Software Competence Center (CCSL) at IME-USP and former director of the Open Source Initiative. He is currently the Editor-in-Chief of the Springer Open Journal of Internet Services and Applications. His research interests include Free and Open Source Software, Agile Software Development, Smart Cities, Data Science, and Distributed Systems.

Paulo Meirelles is an Adjunct Professor at the Federal University of São Paulo (UNIFESP). He is also a Researcher Collaborator at the University of São Paulo (USP), working on behalf of the Free/Libre/Open Source Software Competence Center (CCSL-USP). He was the head of the project of rebuilding the Brazilian Public Software portal between 2013 and 2016. Paulo received both his M.Sc. and Ph.D. in Computer Science from the Federal University of Rio Grande do Sul (UFRGS) and the University of São Paulo (USP), respectively. His research interests include Free Software, Open Source Software, Agile Software Development, Source Code Metrics, and Mining Software Repositories.