Understanding Software Evolution in Multiprovider Scenarios: An Exploratory Study

Anelis Pereira-Vale
Computer Science Department, University of Chile
Santiago, Chile
apereira@dcc.uchile.cl

Tomás Vera[1]
Computer Science Department, University of Chile
Santiago, Chile
tvera@dcc.uchile.cl

Carlos Vásquez[2]
Computer Science Department, University of Chile
Santiago, Chile
cvasquez@dcc.uchile.cl

Daniel Perovich
Computer Science Department, University of Chile
Santiago, Chile
dperovic@dcc.uchile.cl

Jocelyn Simmonds
Computer Science Department, University of Chile
Santiago, Chile
jsimmond@dcc.uchile.cl

Sergio F. Ochoa
Computer Science Department, University of Chile
Santiago, Chile
sochoa@dcc.uchile.cl

Abstract — **The high evolution speed of both the ICT and the user preferences, pushes customer organizations to frequently evolve or replace their software systems in order to keep pace with these evolutions. In the case of custom software systems, their longevity is usually addressed through multiple evolution projects, where each one can be performed by a different provider. In this software evolution scenario, every time a software provider changes, the new provider should gather, as quickly as possible, the knowledge about the product to evolve. Thus, this provider reduces the project risks and tries to get control of the work ahead. Regardless of the evolution effort spent by the new provider, the project results are usually unsuitable, because part of the knowledge of the product structure is lost when the software provider changes. This situation reduces the longevity of the systems and negatively affects the customer organizations. The evolution of custom software in multiprovider scenarios has received little attention from the software engineering community. Therefore, this article presents an exploratory study that analyzes the process conducted by small and medium-sized software providers, to discover and assimilate the knowledge about the product structure of a system developed by another provider. The study explores two particular software evolution scenarios to determine the actors, the structure and dynamic of the knowledge acquisition process, the activities and software artifacts used by the participants, and the effectiveness that providers reach when using these practices and software artifacts. The study also presents several challenges, derived from the study results, which open research opportunities to improve the cost-effectiveness of the product knowledge acquisition process in software multiprovider scenarios.**

Keywords - Multiprovider software evolution, custom software products, product knowledge acquisition, exploratory study, small and medium-sized software companies, product ownership.

I. Introduction

Every day the organizations increase their dependence on the software systems and services used to support their business activities. Therefore, the longevity of their products becomes a highly valuable feature [Mat06, Feb16]. Many of these products are ad hoc (also known as custom or bespoke software), and their longevity depends on the capability of the software providers to evolve them. Recent studies show a high and increasing demand for custom software projects during the last two decades, and this trend is expected to continue [Gvr23, Kbv22]. This high demand for custom projects calls for new mechanisms to evolve bespoke software systems in a more cost-effective way and extend their longevity.

The lifecycle of bespoke products is discrete, i.e., these systems tend to evolve by means of multiple sequential projects (shown as blue triangles in Fig. 1) [Ben02]. In a multiprovider scenario, every evolution project can eventually be performed by a different software vendor. The curve height in Fig. 1 represents the accumulated effort spent in developing and evolving (D&E) a software system throughout its lifecycle.

There are many reasons why these products evolve in a discrete way; for example, after a product is delivered, the customer organization needs time to gather user feedback about the usability and usefulness of the system, identify improvement aspects, and think about the next features of the product. Ozkaya et al. [Ozk10] identify other reasons for this discrete evolution, e.g., to deal with changes in the market, opportunities to adopt new technologies and needs of the stakeholders.

The period between two consecutive evolution projects is uncertain, and it can range from a few months to years. In that period, the provider can change due to several reasons, for instance, it is no longer available or interested in evolving the product, or the customer wants to involve other providers to explore new ideas for the system. In these cases, the product owner (PO), who usually belongs to the provider, also changes [Bja15, Rob21].

[1] Tomas Vera is also with Zenta Group
[2] Carlos Vasquez is also with IdeaUno

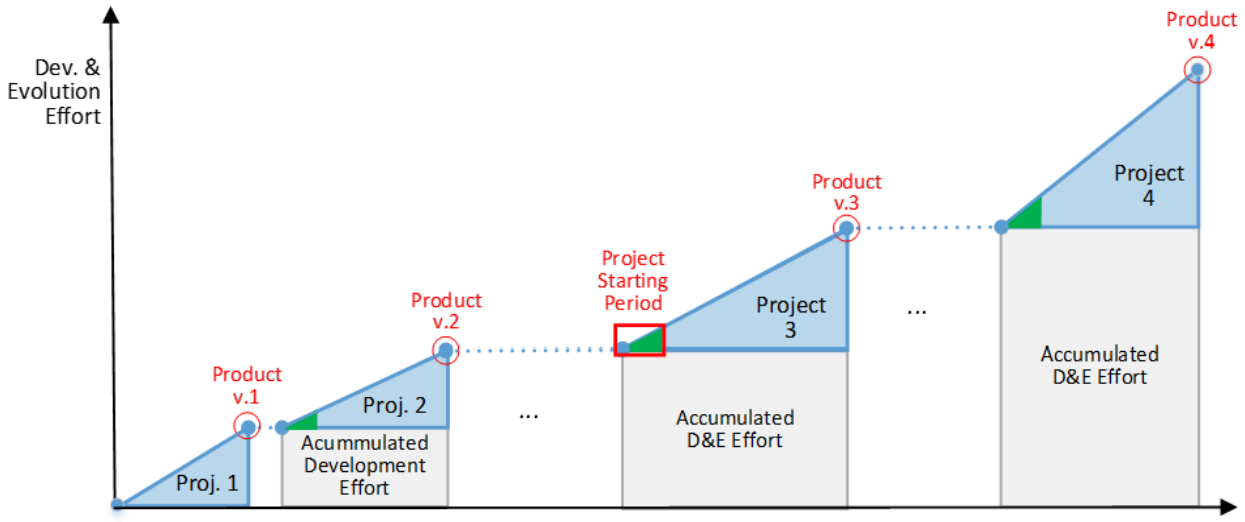

Figure 1. Typical evolution process of a bespoke software product

It may also happen that after a while, the provider continues working on the product evolution, but its former product owner (FPO) is no longer assigned to evolve such a product. There are several reasons for this, for instance, because the FPO left the company or was assigned to another initiative. In any case, the new project leader (i.e., the potential product owner - PPO) and eventually the development team should assimilate the knowledge about the product, as quickly as possible, as a way to reduce uncertainty, identify the challenges to address, and keep the evolution project under control [Ben00]. In order to do that, the participants perform several product knowledge acquisition activities (PKAAs), e.g., product inspections, reviews of software artifacts, and meetings with customer personnel. Most PKAAs are performed at the beginning of the evolution project, in the period known as 'project start' [Sav15a] (this period is shown with small green triangles in Fig. 1). Performing this product exploration at the project start time also allows the PPO to early define a first roadmap to address the evolution project [Ben00].

Addressing the PKAAs in a cost-effective way becomes a major challenge for the PPO and the development team, because the software artifacts that support the product exploration process are not always complete, up-to-date and easy to understand. In addition, performing the PKAAs usually involve time constraints that increase this challenge.

Regardless of the relevance the PKAAs have in the evolution of bespoke software products, they have received little attention from the software engineering community [Fra23], particularly, in scenarios where the evolution projects are performed by different providers [Sne05, Fri12]. Trying to understand both, the process performed by the PPO and the development team to acquire knowledge about the product to evolve, and the effectiveness of such a process, we performed an exploratory study to answer the following research questions:

RQ1: At the beginning of a bespoke evolution project, what software artifacts does the provider review to understand the product to evolve?

RQ2: What activities does the provider usually perform during the product knowledge acquisition stage?

RQ3: How much effort does the provider usually put into performing the PKAAs?

RQ4: After performing the PKAAs, what is the mastery level that the PPO usually reaches?

Answering these questions helps researchers and practitioners identify: (1) the practices that the product owners and their team use to address the PKAAs, (2) the effectiveness of these practices, and (3) the aspects to address to improve their cost-effectiveness. Recognizing the diversity of project evolution scenarios where the FPO changes, in this study we addressed the RQs considering only two particular evolution cases: 1) when the FPO and half of the former development team (or more than that) change, but not the software provider, and 2) when the FPO, the development team, and also the software vendor change.

In both cases, the new project leader (i.e., the PPO) and the development team do not know the structure and features of the product to be evolved; therefore, they have to perform PKAAs from the beginning of the evolution project. Addressing these two cases allows us to identify similarities and differences in how the providers carry out the PKAAs, depending on the evolution scenario. We have focused this study on small and medium-sized software enterprises (SMEs) as solution providers, because they represent the major part of the software industry worldwide.

Next section presents the related work on software evolution, considering a multiprovider scenario. Section III describes the design of the exploratory study, the process structure and the activities performed in each stage. Section IV presents the inquiring instrument used to collect information from the participants. Section V shows the answers of the interviewees, and based on that, Section VI presents the responses to the RQs. Section VII discusses the results and raises several challenges to improve the effectiveness of the PKAAs. Section VIII indicates the threats to validity of the reported results. Finally, Section IX presents the conclusions and future work.

II. Related work

Several research works on software maintenance and evolution have been conducted during the last 50 years [Her13, Sal21]. This stage of the product lifecycle is frequently recognized as the most complex, demanding and expensive [Sto21, Ben00, But22]. To

make the analysis of the related work more understandable, we focus the literature review on the four topics presented in the following subsections.

A. Models to guide the software evolution

An important number of evolution models are available to guide the activities in this stage of the product lifecycle. These models range from Basili's proposals [Bas90] to those recently proposed by ISO/IEC/IEEE [ISO17, ISO22]. Most models prescribe several major activities to perform, and consider the architecture knowledge management as the cornerstone for evolving the systems [Lav11, Ozk10, Sal21]. For using them, the provider should count on comprehensive and up-to-date product specifications, and particularly, with its software architecture of the product to evolve; which is not particularly right in multiprovider scenarios.

Moreover, after analyzing the evolution models proposed until 2017, Lenarduzzi et al. [Len17] indicate that: i) most models were created from scratch without reusing the knowledge of the previous ones, and ii) these models were proposed to address specific evolution challenges in particular application contexts. These contextual aspects also limit the suitability of these models to support bespoke product evolutions in the study scenarios.

B. Architecture-based software evolution

As mentioned before, the literature emphasizes the importance of software architecture in managing systems evolution [Gar09, Kru09, Ozk10, Lav11]. Evidence shows that performing architecture-based software evolution is strongly recommended in any case [Sal21]. However, in multiprovider scenarios the architecture of custom systems is not always available.

When the architecture of the product to evolve is not available, complete or up-to-date, the provider tends to use the source code as a source of truth to identify the product structure and functionality [Men05, Mat16]. The analysis of source code tends to be complex, slow, expensive and error-prone when performed manually. However, it can be performed using software tools that allow the provider to get a visual representation of the code structure (e.g., classes, modules, and programming interfaces) [Ben08, Ber13]. Although these representations are easier to understand than the source code, they also have several limitations, for instance, they represent the source code structure, but not the product architecture. Moreover, trained people are required to understand the visualizations, who are not necessarily available in the provider; particularly, in SMEs. These aspects limit the effectiveness of this approach to support the evolution of bespoke products in multiprovider scenarios.

C. Software evolution based on source code

The research in software evolution based on source code is as extensive as on the product architecture. Depending on the size and complexity of the product, service or feature to be evolved, and also considering the skills of the source code inspector, it can be reasonable to manually explore this software artifact. Low coupling among components increases the chances of performing manual inspections of the source code. However, when the evolution involves several components that are tightly-coupled, it is usually required to use software tools that process source code to generate visual representations of systems (or part of them). These representations do not show the product architecture, but usually the classes in which the source code is organized [Ben08]. It helps developers comprehend the code and identify points to intervene it to carry out the evolutions [Mat16], and also diagnose the software implementation [Ber13, San13].

Their visual appearance of these representations can be ad hoc or adhere to a preestablished metaphor, e.g., the city metaphor [Wet07, Pfa20; Kra22], which have shown to be useful to depict the properties of the source code [Jef19]. Although these visual representations contribute to comprehending the systems implementation, the tools that generate them also have various limitations. For instance, they do not accept (as input) source code written in any programming language, and the visual representations (i.e., the output) become difficult to comprehend when part of the system is implemented using software frameworks.

D. Evolution of custom products in multiprovider scenarios

The product evolution in multiprovider scenarios is diverse, even if we consider only custom products. For instance, in some cases the provider counts on software artifacts to support the evolution process, and in others the vendor mainly has source code to understand the product. When software artifacts are available, they usually are diverse in terms of content, format, and nomenclature. Moreover, the available artifacts are not managed in an integral way, and nobody knows if they are up-to-date [Ras19, Rob21, Sav15b]. This situation challenges the providers, limiting their capability to evolve products and innovate [Fri12].

Frequently, the structural knowledge of the product to evolve is tacit [Raj00, Lap18, Rob21], thin spread, and the knowledge sources are not always available to be asked [Bja15, Rob21]. When people who have the tacit knowledge rotate, or become no longer available to support the product evolution, such knowledge is lost [Hal08, Smo17, Muh22]. Consequently, the product governance decreases, limiting or jeopardizing the next evolution projects.

These reasons make the evolution of bespoke products challenging, diverse and less interesting for software providers than greenfield projects [Sto21]. Therefore, providers are usually reluctant to address brownfield developments, particularly, if the knowledge required to evolve the product is no longer available for the company.

Some research works present general challenges that affect the product evolution in single and multiprovider scenarios. For instance, Mens et al. [Men05] indicate that: 1) there is a need to count on software artifacts specified in languages or nomenclatures more abstract and understandable than the source code, 2) new proposals that provide coherence among these artifacts are required to support the software evolution in an integral way, and 3) more empirical research should be conducted in order to understand in a better way the software evolution process and their different scenarios. In addition, Almashhadani et al. [Alm23] focus on the client-provider relationship during product evolution process, and call for proposals that help: 1) provide manageability, collaboration and communication between both parties, 2) reduce conflicts due to the few documentation available to evolve the products, and 3) facilitate the collaboration and communication among teams during knowledge transfer activities.

Recognizing the contributions of these research works to advance state-of-the-art, we support the invitation to increase our understanding of this particular domain, not only because it has been poorly addressed, but also due to the impact that software evolution and maintenance have on customer and provider organizations. Particularly, we have found no empirical studies that explore the study scenario, and help answer the stated research questions. Next section introduces the design of the exploratory study.

III. DESIGN OF THE EXPLORATORY STUDY

The process conducted to explore the PKAAs in the study scenarios was qualitative and based on interviews. It involved the four stages (Fig. 3): preparedness, semi-structured interview, analysis of answers and closure. In the first stage, we defined and tested the set of questions that guided the interviews, and allowed us to collect information to answer the RQs. The second stage involved interviewing the participants using an inquiring instrument. In the third stage, we performed the transcription of the audio from the interviews video records, coded and analyzed the information, and obtained the elicitations output. Based on these results, during the last stage we answered the RQs and verified these answers performing cross-validation. Next we describe each stage in more detail.

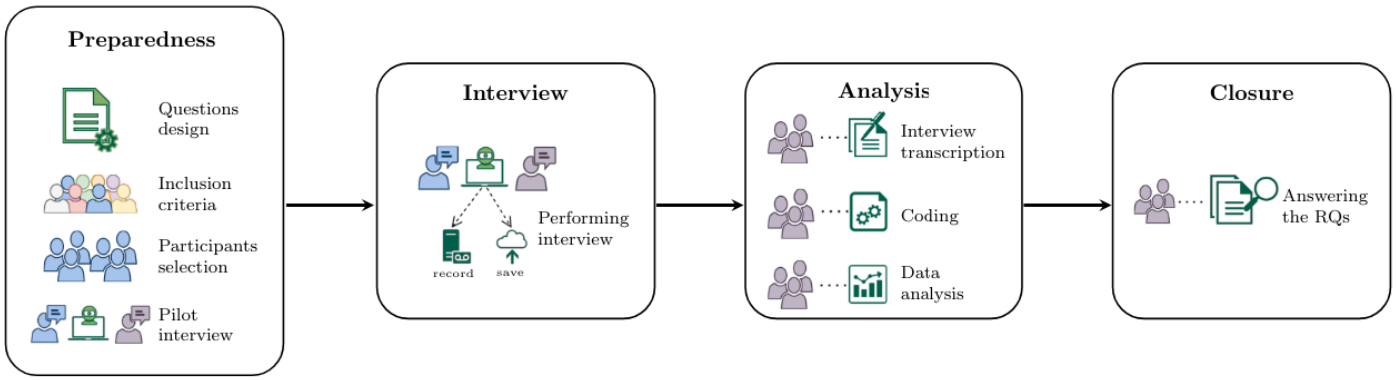

Figure 2. Process performed in the exploratory study

A. Preparedness

This stage considered the following activities: 1) design of the questions to be asked in the interviews as part of the interview process, 2) definition of the inclusion and exclusion criteria for participants, 3) selection of participants, and 4) calibration of the inquiring instrument using a pilot interview.

1) Design of elicitation questions

The design of questions for the interviews (i.e., the inquiry items) was evolving, and it involved four synchronous sessions. Each session lasted approximately 120 minutes and only the authors participated. Considering RQs and following a top-down approach, the research team formulated and agreed 2-3 elicitation questions per RQ. The elicitation questions were designed taking into account that their answers should contribute to responding to every RQ. Then, the authors performed a complementary activity to cross-validate the questions following a bottom-up approach. Particularly, they analyzed if the answers to the elicitation questions would allow them to build and validate the potential responses to the RQs. The process to define and improve the inquiry instrument was repeated until all participants felt comfortable with the output; in this case, such a goal was reached in the fourth session. Then, two external researchers evaluated the inquiry instrument and provided suggestions to make the elicitation questions easy to understand for the interviewees.

The final set of elicitation questions was used during individual semi-structured interviews to the participants, who worked in Chilean and Brazilian software companies. The questionnaire ended up including nine elicitation questions; two of them considered multiple options, and the other six involved open answers. These questions were written in Spanish and Portuguese, and the interviews were conducted in one of these languages depending on the interviewee. This facilitated the understanding of the elicitation questions, the dynamism of the process and the collection of information from the interviewees. All information was translated to English before coding it.

2) Inclusion and exclusion criteria

The inclusion criteria for the software companies were the following: i) to be a small or medium-sized enterprise, ii) to have at least five years of experience developing custom software in multiprovider scenarios, iii) the main core business of the company should be development of custom software.

The inclusion criteria for interviewees considered the following requirements: i) to be working in a bespoke software development company, ii) to be playing a role linked to product development leadership, e.g., product owner, scrum master, or project leader, iii) to have five or more years of experience playing the role, and iv) to have participated in at least three multiprovider evolution projects as leader or development team member during the last five years.

The exclusion criteria considered participants that: i) recurrently evolve the same software products, or ii) new product owners that evolve products using the development team that is regularly in charge of the system.

3) Participants selection

As mentioned before, the software companies involved in this study were small and medium-sized enterprises (SMEs), characterized according to classification reported in [Min14, Bib17]: small (10-49 employees) and medium-sized (50-199

employees). Thirty companies, which accomplished the inclusion criteria, were invited to participate in this study. These companies showed an important maturity in terms of business operation. Some of them were former collaborators of the authors, and other were enrolled in the MITI[3] software association. Typically, these providers perform small evolution projects; however, the products that they address can be medium-sized or large. Most of these providers use Scrum or ad hoc agile processes to address the evolution projects. The main core business of these companies included development and evolution of web and mobile information systems to support several business domains. Although some of them kept developing their own COTS software, most of their activities were focused on developing and evolving custom products in multiprovider scenarios.

Fifteen SMEs accepted the invitation, three of them from Brazil and twelve from Chile. Considering the inclusion and exclusion criteria for the participants, each company proposed up to three candidates to be part of the study. Using this information, a total of 35 people were invited via email, and 21 accepted the invitation. Two of them were selected to be part of the pilot interviews that helped us evaluate the understandability of the elicitation questions, and the other 19 were involved in the study interviews.

4) Pilot interview

During this activity we interviewed the two pre-selected participants to calibrate the inquiring instrument. This allowed us to validate the understandability and relevance of the questions, evaluate the time required to complete the interview, identify possible problems and correct them, as well as to allow the interviewer to gain familiarity with the structure and dynamic of the activity. Given that no major issues were identified during this calibration activity, the inquiring instrument used in the pilot interview was the same as that used to interview the rest of the participants.

B. Interview process

During the second stage we performed the semi-structured interviews, using one session per participant according to the guidelines of Wohlin et al. [Woh12, p. 62]. The interviews were conducted in Spanish or Portuguese depending on the interviewee. The sessions were mediated by videoconference, and recorded after getting the explicit consent of the participants. All sessions had the same structure. We decided to use this type of interview because it usually allows keeping the people engaged with the process, answering questions from the participants, and inquiring on unexpected responses of them [Ful08, p. 14].

At the beginning of every interview, we informed the participants on the goals and dynamics of the activity, and also their right to leave the interview at any time with no consequences for them. Then, we asked for their explicit consent to both participate in the study and allow us to record the interview session.

Before starting the interview session, we asked the participants to suppose that they were the PPO (i.e., the new product owner or project leader) involved in a new evolution project, where they and their development team do not know the product to evolve. Therefore, they have to perform the PKAAs in the two scenarios indicated in the Section I: 1) when the PO and the development team change, but not the provider, and 2) when the provider changes, and the PPO and the development team do not know the product. The participants should answer the questions considering their role and these evolution scenarios. The same interviewer guided all sessions that lasted 50 minutes on average.

C. Data analysis

Once collected the information, the analysis of the answers involved three activities: 1) transcribing the interview records, 2) coding the responses, and 3) analyzing the coded information to obtain initial answers to the RQs.

D. Closure

During the closing stage, we used the qualitative information given by the participants to perform cross-validation of the initial answers to the RQs. This helped us get a more in-depth understanding of the answers, contextualize them and identify the findings of this study.

IV. Inquiring Instrument

As mentioned before, 2-3 elicitation questions were defined for every RQ, ensuring that the answers to the former allowed us to build and validate the answer to the latter. Next, we present the elicitation questions related to each RQ:

RQ1: At the beginning of a bespoke evolution project, what software artifacts does the provider review to understand the product to evolve? Answering this RQ helped us identify the explicit knowledge that remains available and useful after changing the FPO.

1. *At the beginning of a bespoke evolution project, what software artifacts do you usually receive to perform the system evolution*? The answers to this question allowed us to identify the explicit knowledge that is regularly available at the "project start" period.

2. *What is the perceived quality of these artifacts considering their completeness, up-to-datedness and understandability?* Options: high, medium, low according to the participant perception. This helped us realize the perceived usefulness of the artifacts received to support the new product evolution project.

RQ2: What activities does the provider usually perform during the product knowledge acquisition stage? The answer to this question allowed us to discover the structure and dynamic of the product knowledge acquisition process (if any), the effort spent on it, and determine if the provider has a regular strategy to address the PKAAs.

[3] WWW.MITI.cl

3. As PPO, *what activities do you and your team perform to analyze the software artifacts?* This helped us realize the ways of retrieving the explicit knowledge from the software artifacts, and eventually the effort spent, and also required, to conduct these PKAAs.
4. *What activities do you and your team usually perform to gather the tacit knowledge about the product?* This question has a purpose similar to the previous one, but it is focused on gathering the tacit knowledge.
5. *How do these activities are performed over time?* The answers allowed us to understand the dynamic and the periods in which the PKAAs are performed during the evolution projects. The answers also shed light on the effort required and spent to deal with these activities.

RQ3: How much effort does the provider usually put into performing the PKAAs? The answers made us realize the effort regularly spent by the providers. Moreover, it helps us determine whether current PKAAs can be considered cost-effective and also cross-validate the responses to questions 3-5.

6. *How many hours a day or week do you usually spend in performing these activities?* More than identifying absolute values, this question explores whether there is a time quantum of the labor journey assigned to the PKAAs and how important that quantum is.
7. *What is the timespan available to complete these activities?* It helps determine if there is a deadline to complete the PKAAs, and how long is the product exploration period. If there is not a formal deadline, then these answers help us identify the criteria used by the PPOs to consider the PKAAs as completed. If there is a deadline, then the answers allow us to realize if the exploration activities are performed under time pressure.

RQ4: After performing the PKAAs, what is the mastery level that the PPO usually reaches? The answer to this question, combined with the response to RQ3, allowed us to determine how cost-effective are the practices currently used to address the PKAAs.

8. *After performing the previous activities, what is the level of mastery that you usually reach on the product?* Options: i) similar to a product owner, ii) basic knowledge on the product, and iii) no control on the product. This allowed us to determine the most frequent outcome of the PKAAs from the PPO perspective.
9. *In your experience, what are the main limitations in quickly acquiring knowledge on the structure and functionality of the product?* Answering this question helped us identify sources of problems and opportunities to improve the performance of the PKAAs.

V. Study Results

The interviews results showed that 10 out of 19 participants were mainly involved in evolution projects performed in the first scenario (i.e., when the FPO and PPO belong to the same provider), five participants worked almost exclusively in projects conducted in the second scenario (i.e., when the FPO and PPO are part of different providers), and the other four participants worked in both scenarios. Next, we present the answers to the elicitation questions for both scenarios.

A. *At the beginning of a bespoke evolution project, what software artifacts does the provider review to understand the product to evolve?*

Considering the participants answers, Figure 3 shows the software artifacts available to support the PKAAs in both scenarios. It also shows a percentage of expected availability for each artifact at the beginning of a custom project; i.e., the feasibility of the provider to count on each particular artifact to perform the PKAAs. These percentages were rounded to the closest integer value. As expected, the source code and the user interfaces of the system are regularly available for the PPO and the development team. Various interviewees indicated that, in the second evolution scenario, the demo of the product (i.e., the review of user interfaces and user journeys) accelerates their understanding of the product functionality.

Unsurprisingly, the legacy information to support evolution projects in the first study scenario is more extensive than in the second one. Excluding the source code and user interfaces, in scenario 1 the provider frequently counts on some representation of the current product architecture, its software requirements (or list of services that it provides) and general information of the system (e.g., supported user profiles, and configuration and deployment information). Less frequently, the provider counts on the data model and roadmap of the product, or a backlog left by the former team.

In the second scenario, the provider mainly has the product's source code and user interfaces as up-to-date information, and eventually some other complementary information, like a list of requirements, a chart of the system architecture and general information of the system. Some interviewees indicated that they demand more than source code and user interfaces to the customer, before accepting to perform an evolution project in this second scenario.

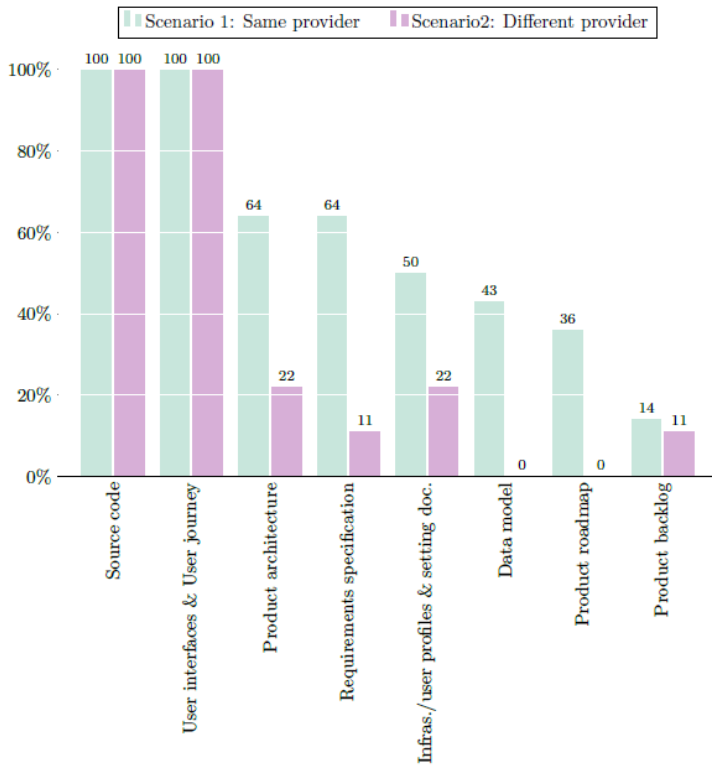

Figure 3. Software artifacts used to support product evolution projects

B. What is the perceived quality of these artifacts considering their completeness, up-to-datedness and understandability? Options: high, medium, low.

The quality of the software artifacts influence their usefulness for evolving the product. In this case, we defined the quality of an artifact based on the interviewees perception about the completeness, up-to-datedness and comprehensibility of such an artifact. Figure 4 shows the average perceived quality, per artifact and study scenario, according to the participants' opinion.

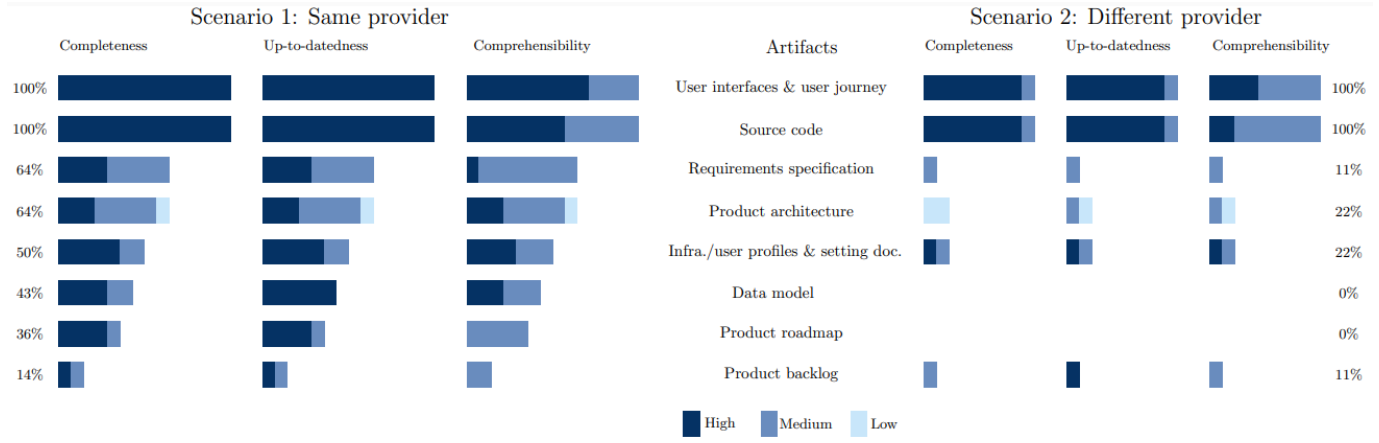

Figure 4. Perceived quality of artifacts according to the participants.

The answers indicate that the providers have access to the source code and screenshots (or demos) at the beginning of the project in both scenarios. Because the system to evolve is usually running in a testing environment or into production, the level of completeness and up-to-datedness of these software artifacts is frequently high. However, some participants indicated that the PKAAs are not always performed using the latest version of these artifacts, mainly in the second scenario.

Concerning the comprehensibility of these artifacts, the user interfaces and the system navigation are more understandable for the PPO than the source code. The understandability of these artifacts in scenario 1 tends to be higher than in scenario 2, because the artifacts generated by the same provider, even when they were created by different teams, tend to follow similar patterns or styles. In addition, the participants indicated that performing tutorials, demos (user journeys) or similar activities considerably eases the understandability of these artifacts.

Regarding the exploration of the source code, it is usually done by the PPO and skilled members of the development team. Therefore, the comprehensibility of this artifact depends on both the individual skills of the inspectors and their ability to perform joint work. However, the participants indicated that having the opportunity to ask questions to former developers of the product makes a difference in the understanding of the source code. This situation is evidenced by comparing the comprehensibility of the

source code in scenarios 1 and 2 (Fig. 4). The same happens when the new team intends to comprehend any other artifact that describes the product, not only the source code. Various participants reported the use of reverse engineering tools to help understand the structure of the source code, but it is usually not part of a regular policy of the providers to understand this artifact.

Figure 4 also shows that several pieces of information and artifacts tend to be lost when the provider changes, e.g., the sketches of both the system architecture and data model, or the product roadmap. According to the interviewees, it usually happens because such artifacts are managed in repositories of the provider, and used to support the internal activities of the development team. Typically, this information is partially or not delivered to the client at the end of the project, because it is not usually part of the contract. Because most customers are not conscious of the relevance of this information in the future evolutions of the product, they do not include it in the project contract.

Concerning the data model of the systems, it can be retrieved from the current database of the product, using software tools that perform reverse engineering. However, various participants indicated that when the system implementation uses data management capabilities provided by the modern software development frameworks (e.g., Django), the output of the reverse engineering process is usually not easy to understand. This happens because the resulting data model specification includes components and relationships that are required for the proper operation of the framework, but increases the complexity to comprehend such an artifact.

Regarding the product roadmap, it usually does not exist; and if there is one, such specification is kept at the provider side and not transferred to the customer. Therefore, in scenario 2 such information is lost after the first evolution project. Coincidentally, the participants did not report product roadmaps governed or stored by customer organizations nor the existence of product owners in that side.

C. *As PPO, what activities do you and your team perform to analyze the software artifacts?*

The participants indicated three non-exclusive activities that are usually performed to analyze the software artifacts: 1) meetings with the customer personnel, 2) meetings with the former project leader or team, and 3) autonomous review of the artifacts. The way to perform the last activity is diverse, e.g., in some cases only the PPO analyzes the artifacts, and in others cases, the activity becomes more collaborative, where team members analyze particular artifacts and then report the results to the PPO or the team.

The autonomous review is the PKAA most frequently used in scenario 1 (Fig. 5). Typically, it happens because the former team shares the internal information of the product with the PPO and the new developers. This analysis of legacy information is usually complemented with clarification meetings conducted with the FPO, project leader, and technical personnel of the former team. The participants indicated that the PPO usually meets with representatives of the customer to validate assumptions and expectations about the product being evolved.

In scenario 2, the PPOs and their team frequently use meetings with the customer personnel, as a first step to get a big picture of the product functionality and major components. Then, they perform autonomous reviews of the source code and user interfaces, given that not much extra artifacts are available for them. According to the participants, the meetings with the client personnel are mandatory in this scenario. They also indicated that the former PO and development team are usually not available for questions of the new provider. This increases the complexity of performing the PKAAs and the loss of knowledge about the structural design of the product.

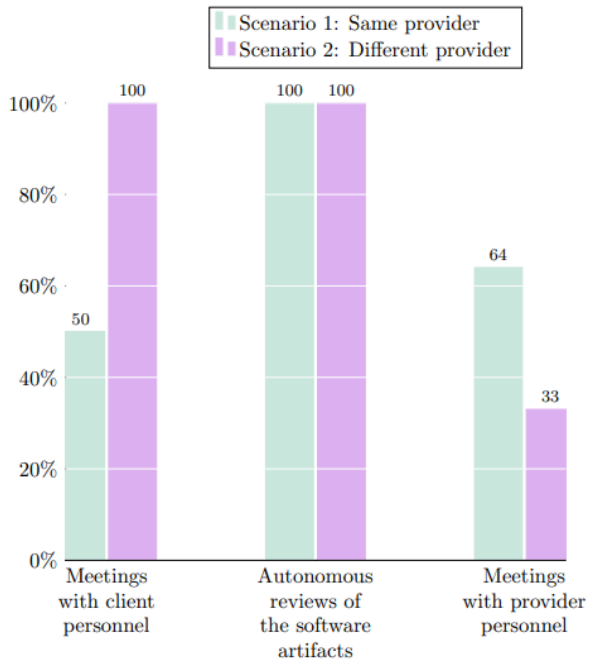

Figure 5. Product knowledge acquisition activities conducted by the PPOs

Figure 6 shows these PKAAs in more detail. Because the participants used different names for referencing the same activities, the similar ones were grouped and labeled with a single name to properly compute and report them. Particularly, the *exploration meetings* and *interviews* are used by the provider to gather information from the client on several aspects of the product, e.g., the user profiles being supported and the functionality available for them.

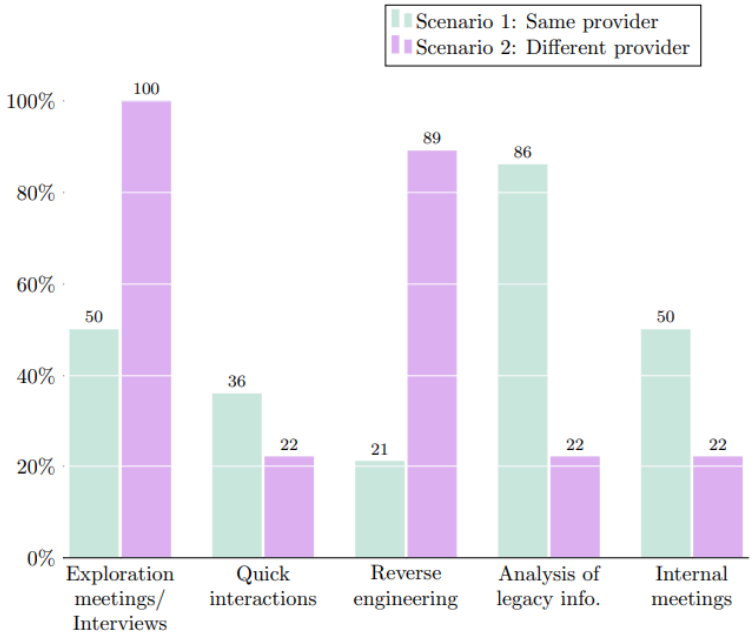

Figure 6. Detailed activities to perform the product knowledge acquisition.

On the one hand, in these exploration meetings the customer personnel inform the provider on particular aspects of the product; it mainly happens in scenario 2. These meetings are among the first ones to be performed in the product exploration process, because it helps the PPO and new team build an initial shape of the product, and also synchronize expectations with the client. Typically, each exploration meeting lasts 1-2 hours, and a small evolution project usually requires one or two of these meetings.

On the other hand, the interviews are also meetings, but shorter than the previous ones. They are usually focused on getting a more in-depth understanding on product components and features, and customer expectations. Typically, the interviews are semi-structured and guided by the PPO.

Similar to the previous activities, the *quick interactions* also involve the provider and the customer. They are informal (unplanned) and usually performed by email, instant messaging or phone calls. These interactions are used to ask the customer for specific issues that can be answered in short (e.g., information confirmation or clarification) and help the provider advance in the product knowledge acquisition. When the provider's questions involve information that is critical for the evolution project, the PPO usually complement these interactions with a formal confirmation of the customer's answer. Considering the interviewees comments, no major difference was identified in the use of this practice in scenarios 1 and 2. The differences shown in Fig. 6 could respond to the fact that in scenario 1 there usually are more people from the former project to interact with, and also more artifacts that make the development team raise questions.

The *reverse engineering activity* involves the analysis of source code to obtain more abstract representations (usually, visual representations), which help providers understand the structure of the system implementation. This activity can be done manually (using code inspections) or supported by external tools that process the source code and generate these representations. The literature reports the use of this practice in evolution scenarios where the product specifications tend to be insufficient [Sch10, Tri15, pp.10], as it usually happens in scenario 2. Moreover, in this scenario the former provider is usually not available for answering questions of the new provider, therefore, the use of these practices becomes more relevant than in scenario 1. In this second case, the availability of members of the former team to support the new evolution project, reduces the need to perform these reverse engineering activities.

Concerning the *analysis of legacy information,* it includes the review of other software artifacts by the provider personnel, for instance, requirements specifications or architectural charts when available. This activity is most frequently used in scenario 1 due to more software artifacts are available than in scenario 2.

Regarding the *internal meetings,* most participants indicate that these activities are not regular, but performed on-demand when required; the PPO usually decides when to do them. In these meetings participate the PPO and members of the development team, and eventually members of the former team in scenario 1. The goals of the internal meetings are diverse, for instance: 1) to build a shared understanding (or a diagnosis) about the product's structure and functionality, 2) to maintain communication and coordination between team members while they perform the PKAAs, 3) to identify and make visible the challenges, risks and obstacles to evolve the system. When the evolution project happens in scenario 1, the internal meetings involving members of the former team become valuable to get contextual information for the new project, for instance, the level of technology appropriation of the users and commitment of the counterpart, or the regular alignment among the stakeholders expectations.

The internal meetings are also performed in scenario 2, but less frequently than in scenario 1. The reasons seem to be similar to the use of quick interactions; i.e., there are fewer artifacts to explore and, therefore, less sharing knowledge and coordination activities are required. Moreover, no meetings with the former team are usually conducted in this scenario, because the former provider is usually not available for the new provider.

D. What activities do you and your team usually perform to gather the tacit knowledge about the product?

The participants indicated that they do not perform particular activities to gather the tacit knowledge about the product. All activities involving the customer´s personnel or the former development team are used to gather and validate the tacit and explicit knowledge. Various participants highlighted that making explicit the tacit knowledge, and making the new team aware of it, is highly important, but represents a major challenge for the new provider. Therefore, if such an activity is not part of the contract, it is usually not performed at least that the provider has a good reason to do it; for instance, when this vendor is sure that it will be in charge of evolving the product for a long time period. If the tacit knowledge of a product is not specified in some software artifact, the evolution process will keep its current status-quo, and it will be difficult to improve the cost-effectiveness of the PKAAs. Considering the evolution scenarios explored in this study, the projects conducted in scenario 1 usually involve less tacit knowledge than those performed in scenario 2; it is evidenced in Fig. 4. Therefore, less activities with that purpose are also performed (Fig. 6).

E. How do these activities are performed over time?

The participants indicated similar answers to this question for both scenarios. Typically, they perform an *initial exploration* process, and then various *focused inspection activities* during the project. The initial exploration is conducted at the project start period, and it is frequently addressed using a loosely-coupled approach [Pin05]; i.e., this exploration combines periods of autonomous work of the PPO and teammates (e.g., analysis of legacy information), and sporadic knowledge synchronization activities (e.g., internal meetings).

The PPO coordinates the process and assigns specific exploration responsibilities to the team members. The results of individual explorations are reported to the PPO or to the whole team during the knowledge synchronization activities. Thus, the PPO and technical leader get an overview of the product and client expectations. This allows them to gain self-confidence to explore particular aspects of the product, and determine how to evolve it. Two participants indicated that they usually intend to perform a formal initial exploration activity at the beginning of the project (implemented as a sprint 0), because it eases and accelerates their understanding of the product. However, the feasibility to carry out such an exploration depends on a negotiation with the customer who pays for it.

Concerning the *focused inspection activities* performed during the project, most participants (17 out of 19 interviewees) indicated that they are conducted on-demand, mainly by the developers, when required. In scenario 1, these activities include not only the analysis of legacy information, but also interactions with the current and former development team. In scenario 2, it is limited to the analysis of legacy information and eventual interactions with the teammates.

The focused inspections are probably a consequence of using Scrum as development approach, which focuses the providers on understanding the part of the system involved in the current or next sprint. Consequently, different parts of the product are inspected in different periods (e.g., sprints or increments), sometimes involving different people. This situation produces thin spread knowledge that usually remains tacit during the whole project; therefore, such knowledge is prone to be lost when there is turnover of personnel in the development team, or after the project closure.

Depending on the granularity and relevance of the artifacts being explored, the developers share the results of the focused inspections with the PPO, similar to the initial exploration. Considering the interviewees comments, no major differences were identified in the way of performing the activity in the studied scenarios.

Summarizing, most participants indicated that they spend an important effort in conducting the PKAAs at the beginning of the project, and then they continue performing these activities on-demand during the rest of the project; usually, in an informal way and focused on particular parts of the system. The focused inspections generate mainly tacit knowledge that is lost when the evolution project finishes.

F. How many daily or weekly hours do you usually spend in performing these activities?

The participants indicated that there is no predefined time allocated to perform the initial exploration or the focused inspection activities. Typically, during the initial exploration, the PPO (as coordinator of this activity) assigns a deadline and a goal to particular members of the development team. These people conduct the PKAAs that they consider appropriate to reach the goal, and use several time windows during the labor journey. Therefore, the amount of time spent in this scattered process is uncertain and depends on several variables, e.g., the availability and skills of the people to explore artifacts, the complexity and size of such artifacts, and external help received to address de PKAAs.

Various PPOs indicated that they usually self-assign an *internal deadline* to get the basic knowledge of the product, but such a deadline is not always visible for the development team. The internal deadline is defined considering the formal deadlines agreed with the client to deliver the new features or perform the adjustments to the product. Therefore, the amount of time spent in the initial exploration activity continues being uncertain, but constrained by the internal deadline imposed by the PPOs. These PPOs also indicated that the PKAAs can be finished before the internal deadline, when they feel that the product evolution can be done under control with their current understanding about the system.

Concerning the focused inspection activities, they are shorter than the initial exploration. The time allocation for each inspection is also uncertain and considers informal deadlines. The deadlines are usually assigned by the person in charge of the activity, and then agreed with the PPO.

The interviewees also indicated that the knowledge acquisition speed is considerably higher in the scenario 1 (same providers), because not only more software artifacts are available, but also former team members are usually available to support the new team. However, the time spent in the PKAAs in scenario 1 is not necessarily shorter than in scenario 2.

Concerning the two participants that indicated to perform a sprint 0 to address the initial exploration (whether it is feasible), they mentioned that such sprint usually lasts one or two weeks, depending on context variables like the size and complexity of the product to address and evolution to perform. Typically, people participating in the sprint 0 have full dedication to the PKAAs; however, the number of the participants and their skills depend on what is required in each evolution project. It is important to remark that the SMEs involved in this study mainly perform small evolution projects; therefore, the answers of the participants should be understood under that context.

G. What is the timespan available to complete these activities?

In most cases, the answer to this question was indicated in the answer of the previous one. Summarizing, the timespan to address the initial exploration activities is arbitrary, self-defined by the PPO, and its length usually depends on the size and complexity of the artifacts to be explored and the level of understanding required to evolve the product. The time window to perform the rest of the exploration activities (i.e., the focused inspections) is flexible, and it can be adjusted by the PPO according to the results already obtained in the previous PKAAs. The timespan for focused inspections is uncertain, usually managed by the person in charge of performing them.

The participants indicated that in particular situations, e.g., when the provider is sure that its company are going to perform several evolution projects on the current product, the PPO usually assign people to understand in depth and specify key software artifacts when they are not available; e.g., the system architecture, data model or major software services. Thus, the provider reduces risks to address the next evolutions of that system. Typically, there is not a formal timespan to complete these activities, and according to the participants, they can last from some few weeks to various months. The use of long time periods to perform this exploration is usually a symptom of lack of artifacts that help understand the product, information decentralization, or lack of a PO or someone else playing such a role.

H. After performing the previous activities, what is the level of mastery that you usually reach on the product? Options: i) similar to a product owner, ii) basic knowledge on the product, and iii) no control on the product.

According to the participants, the level of mastery that is reached is uncertain because it depends on several variables, for instance: i) the type of product addressed (size, complexity, etc.), ii) time windows available to learn about the product, iii) the type and quality of software artifacts that are available, and iv) the external help received in the exploration process. Frequently, the level of mastery reached by the PPO usually tends to be enough to evolve products in scenario 1, but insufficient for doing it in scenario 2 (Figure 7). In projects performed in scenario 2, the PPO usually relies on the knowledge and capability of the development team to manage product evolutions under uncertainty.

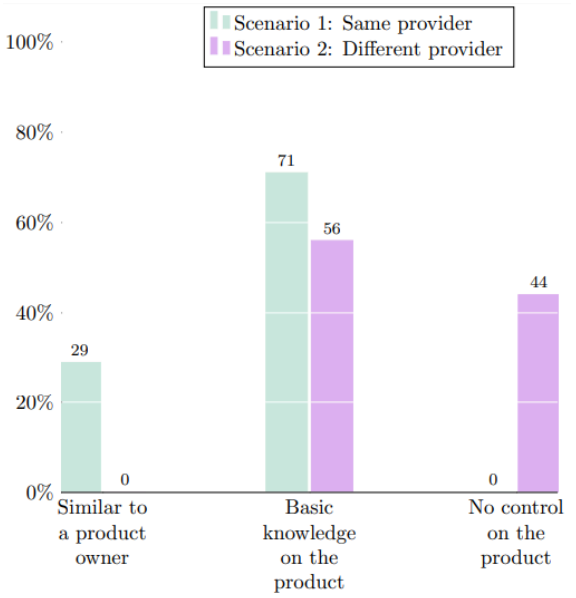

Figure 7. Mastery level reached by the PPO after performing the PKAAs

Various PPOs mentioned that they do not intend to become the PO during an evolution project, because they do not know if they will participate in the next evolutions of the product. Instead, they intend to gain enough knowledge related to the aspects to evolve according to the contract, and thus perform the evolution project in a controlled way. As mentioned before, if the PPOs

identify a high probability to continue evolving the product after the current project, they usually spend an extra effort to gain control on the product. According to the participants, this attitude is frequently observed in both scenarios.

I. In your experience, what are the main limitations to quickly gain knowledge on the product structure and functionality?

The participants tend to agree that the size and complexity of the product is the primary limitation to quickly understand the system structure and functionality. However, they also indicate several other limitations to perform the activity, for instance: i) the availability of relevant and easy-to-understand information, ii) the quality and granularity of the supporting information, iii) the time available to perform the reviews, iv) the quality of the external support received by the PPO and the current team, v) the low motivation of the involved people, and vi) the differences among the PPO, client and development team on their understanding of the product. Next we introduce each of them.

Availability of relevant and ease-to-understand information. It is quite clear that the availability and relevance of the supporting information play a key role in the product knowledge gathering process. However, various participants highlighted the effort required to understand and assimilate such information, as a major limitation to quickly identify the product structure and functionality in a holistic way. According to them, this effort is influenced by several factors, like the amount of explicit and tacit knowledge that is available, how scattered and visible is the tacit knowledge, the understandability of the software artifacts specification, and the availability of external people who can help understand the product and its artifacts.

Quality and granularity of the supporting information. In general, there is little formal information related to the product, and such information is difficult to consume (e.g., source code, little/no connected documents, or documents with any structure and content). Typically, organizing these pieces of information to generate a holistic view of the product considering different concerns (or aspects), is a difficult and time consuming activity. Among these concerns the participants mentioned the *operational environment* and the *system structure*. The first one includes the components of the ecosystem (usually, a context diagram or similar), the instructions to configure and deploy the product, and the technologies used in the system implementation. The second concern includes components of the software system, for instance, the software architecture, the containers, the software integrations (APIs to provide or consume services) and the data model. Furthermore, documentation varies from project to project, provider to provider, even when the same development method is used.

On the other hand, the quality of the artifacts received by the PPO is usually average in terms of completeness and understandability. In projects that use artifacts created by the same provider (e.g., in scenario 1), these artifacts tend to be more complete and understandable, given their creators and readers are more familiar with the structure and nomenclature used in the artifacts specification. Some providers ask the development team for completing the product documentation before closing the project, which usually generates a gap between what is implemented and what is documented. This gap is usually detected in the next evolution projects. Regardless of the obvious need of keeping the system documentation complete and updated, in practice such information is not perceived as an asset that both provider and customer have to take care of it.

Available time to perform the reviews. Various participants indicated that the PKAAs are frequently addressed during the few free time of the involved people, after performing their mandatory activities. It means that the PO and developers tend to perform such activities when they are fatigued, which negatively affects the quality of activities' output. In addition, the length of the time windows for reviews is more related to people availability, than to the time required to properly carry out the exploration of the artifacts.

On the other hand, there usually are schedule limitations to organize synchronous review activities between the PPO and team members, which jeopardizes the knowledge sharing and coordination among the participants. Consequently, the capability to allocate enough quality time to perform the PKAAs is compromised, and this negatively impacts on the effectiveness and efficiency of the software product evolution project.

Quality of the help received by the PPO and the current team. Depending on the size and complexity of the evolution project, the PPO could require help from members of the development team, and also from people involved in previous evolutions of the product, e.g., the former PO and developers or client personnel. The need for this help ranges from "not required" (in small evolution projects) to "highly required" (in large evolutions). Various participants indicate that, when this support is required, the quality of the external help makes a difference on the output of the PKAAs. While good support usually accelerates the knowledge acquisition, inappropriate support does waste time, limiting the capability of the PPO and the team to explore the product on-time. In addition, the skills of the people who provide and receive this help also make a difference on these activities' outcome.

Low motivation of the involved people. Although these product exploration activities are vital for project success, the participants mentioned that no one wants to do them because of several reasons. First, the PKAAs are knowledge intensive, boring, time-constrained and doing them properly usually requires an important individual and collective effort. Second, these activities are usually perceived by the involved people as secondary since they distract the people from their main activity, which is to build the new solution. Third, the effort spent in these activities is usually not recognized by the client or provider organization. Fourth, no one knows if the effort spent on performing the PKAAs is worth, because this could be the first and last evolution project for the current provider; moreover, everyone is aware of it. Consequently, the willingness to participate in the PKAAs is usually low; it considers the members of the new and former development teams, the PPO and also the stakeholders. In the same line, the motivation to improve the information that supports the PKAAs is also low. This tends to keep the status quo of these activities when performing the next evolution project. Evidently, this limitation is much more harmful in scenario 2 than in scenario 1.

Differences in the product understanding among the involved people. According to the interviewees, the thin spread information and tacit knowledge about the product, and also its limited observability, tend to produce differences on what is and is not the product according to the participants (i.e., the PPO, the development team and client counterpart). These discrepancies usually slow down the process of identifying a common and accurate starting point for the evolution project.

VI. ANSWERS TO THE RQS

The answers to the elicitation questions allowed us to not only build the answers to the RQs, but also understand the similarities and differences between both study scenarios. Next, we present the responses to the RQs.

A. *RQ1: At the beginning of a bespoke evolution project, what software artifacts does the provider review to understand the product to evolve?*

Figure 8 illustrates, using blue squares, the answer to this question considering the evolution scenarios. As we can see, the provider usually counts on the source code and the user interfaces to start its exploration and the evolution of the product. Then, in scenario 1 the new team frequently counts on additional artifacts mainly for two reasons: 1) they have access to several documents and specifications that were used internally for the former team, 2) after a couple of evolution projects the providers feel that they probably continue evolving the product. This second reason pushes the provider to generate more and better artifacts that remain available for the next evolution project addressed by them.

In the case of scenario 2, the availability of artifacts is lower than in scenario 1 due to several reasons, such as: 1) the tacit knowledge of the former team was never made explicit, 2) the former team transferred to the client only the artifacts indicated in the contract, and 3) part of the transferred artifacts were lost by the customer, because usually nobody from the client side takes care of this information; i.e., there usually is no formal guardian of the software artifacts in the customer organization.

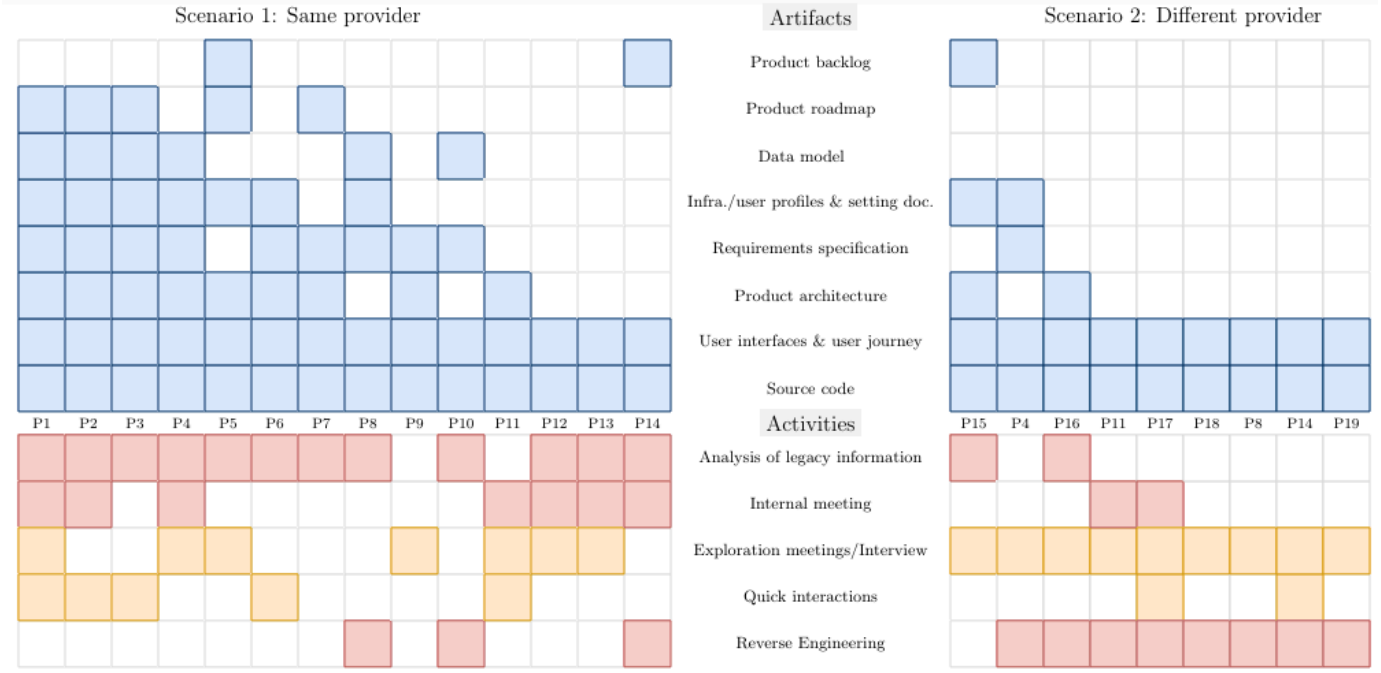

Figure 8. Artifacts and activities used in the product exploration process.

Concerning the use of the artifacts to support the PKAAs, it is important to consider that the acquisition of product knowledge involves two stages, which are typically performed in sequence: 1) to understand the product in a holistic way, and 2) to understand the specific aspects or features involved in the evolution project. The artifacts required to support each stage are different. In that sense, every PPO or team member in charge of an activity decides what artifacts to use, according to their own criteria and the feasibility of using them. Therefore, these software artifacts should be perceived as tools in a toolbox, which are available to support both stages of the product knowledge acquisition.

According to the interviewees, there are no useless artifacts; however, some of them are perceived as more valuable than others depending on the product exploration stage being addressed. For instance, the software architecture, data model and product roadmap are recognized as highly valuable to understand the product in a holistic way (i.e., the first stage of the process). To support the same stage, the system configuration and deployment instructions (included in the category "infra./user profiles and setting documentation" in Fig. 8) are recognized as valuable to reduce the effort of the provider to make the system run in order to explore it mimicking the users.

The completeness, up-to-datedness and understandability of the artifacts also influence how much they are used in-practice. In particular, low-quality artifacts are kept away from the shared repository of the project to avoid team members wasting time analyzing these documents that lead to misunderstandings.

B. *RQ2: What activities does the provider usually perform during the product knowledge acquisition stage?*

The answer to this question, illustrated using orange and red squares in Figure 8, is similar to the previous one in the sense that these activities should also be considered as tools for addressing the product exploration process. The PPO and the team decide to use a particular activity considering their expertise and availability to perform it, the artifacts and external people available to support it, and its suitability to address the exploration goal at the current stage of the product knowledge acquisition process. The differences in the frequency of use that we see between both scenarios are directly related to it (Fig. 8). For instance, in scenario 1 the providers mainly perform analysis of legacy information because the software artifacts and external people required to support such activity are usually available. In scenario 2, where the availability of artifacts and external people is clearly lower, the providers have to perform interviews and reverse engineering to try to gather the same knowledge as in scenario 1.

Typically, these activities are intensively used at the beginning of the project (i.e., during the *initial exploration)* and then performed on-demand in the *focused inspections* during the whole project. Therefore, the PKAAs end-up happening almost in parallel with the implementation of the product evolution. This is aligned with the way to evolve a product using software process agile approaches.

C. *RQ3: How much effort does the provider usually put into performing the PKAAs?*

The effort invested to perform the PKAAs is uncertain in both evolution scenarios, given that it depends on several factors such as: i) the size and complexity of the project, ii) the capabilities of the PPO and the team, iii) the availability and quality of software artifacts, and vii) the amount and quality of external support that the new team receives. Most participants indicated that they spend effort in conducting the PKAAs usually until reaching one of the following situations: 1) the team already gathered the knowledge required to keep under control the current evolution, and 2) the PPO indicates that there is no extra time to continue exploring the product. However, if the provider thinks that there is an important chance to continue evolving the product after the current project, then it usually assigns extra time and people to explore the product, even to make explicit part of the tacit knowledge.

Due to the differences in the available resources and external support received to perform the PKAAs in both study scenarios, the participants believe that they spend less effort in exploring the product in scenario 1 than in scenario 2. Moreover, they also perceive that they end-up getting a more in-depth understanding of the product in the former scenario than in the latter.

D. *RQ4: After performing the PKAAs, what is the mastery level that the PPO usually reaches?*

The results shown in Figure 7 indicate that teams are considerably more cost-effective performing PKAAs in scenario 1 than in scenario 2. This is a finding that can also be envisioned from the responses to RQ3.

The study results also show that in most cases, after performing the PKAAs, the PPO and the development team reach a basic knowledge about the product. This is a consequence of our poor practices, artifacts and motivations to perform these activities. The limitations, already presented in section V.I, open several research opportunities to improve the cost-effectiveness of the product exploration process in multiprovider evolution scenarios.

VII. Discussion

These results illustrate the complexity and the challenges of conducting the PKAAs in the study scenarios. They also show that the software evolution scenarios are diverse, particularly in terms of artifacts and people available to support the product exploration process. This support has a direct impact on the outcome of the PKAAs, which is usually poor. Next, we discuss how several aspects of the product exploration process can be addressed to improve its effectiveness in the study scenarios.

Concerning the structure and dynamic of the product exploration process

This process is informal and usually slow, knowledge intensive and time-constrained. In the studied scenarios, its structure tends to involve two stages: an initial wide exploration of the product, and then several on-demand focused reviews that are performed individually by the team members. Consequently, the exploration process is usually scattered and therefore it requires coordination, usually by the PPO, to ensure an appropriate knowledge integration and sharing. In most cases, the exploration process requires first to collect thin knowledge, integrate it and sometimes to make it explicit before sharing it to the development team.

Regardless of the complexity and relevance of this process, the provider does not allocate formal time to explore the product, and the participants do not receive recognition for getting a suitable outcome from this process. Consequently, the development team and external people tend to be reluctant to participate in these processes, except when the evolution is small and focused, or represent an important opportunity for the provider. Therefore, a first challenge for practitioners and researchers is *to conceive new proposals that reduce the complexity of this process and make it valuable for the participants*. That will make the product exploration process more effective.

Concerning the information supporting the process

The scenario in which the product evolution is performed usually influences the availability of information and people to support the PKAAs. Concerning the supporting information, it involves software artifacts (explicit knowledge) and also tacit knowledge that are not always easy or feasible to access by the new providers. Typically, the explicit knowledge on the product is kept in a shared repository by the development team, as a collection of thin spread artifacts. Such information is usually shared with other teams of the same provider, but no with teams of other providers. Moreover, that information is not delivered (or partially delivered) to the customer organization depending on what was specified in the project contract.

When the software provider changes, these pieces of information and the tacit knowledge get lost if they are not delivered. Even when delivered to the customer, it frequently gets lost if the such organization has not a custodian of that information. Therefore, the next evolution project of that product tends to start in similar conditions than the previous ones. In this sense, a second challenge for researchers and practitioners is *to propose new mechanisms to reduce the loss of tacit and explicit knowledge when the provider or an important part of the development team changes*. To properly address this challenge, it is required to change the role of this supporting information in the evolution processes. Instead of providing only a technical view of the current system, it should represent the product roadmap that allows customers and providers to govern the product evolution throughout its lifecycle. This will increase the perceived value of this information, and therefore, there will be more commitment to take care of it and keep it up-to-date.

The software artifacts, even when available, are not always easy to understand and relate to the new provider. This makes the participation of external people, e.g., the former PO or development team, be frequently required to get a holistic view of the product in short-time. Unfortunately, these people are not always available or willing to participate; therefore, it is required *to conceive artifacts representations that are easy-to-understand and interrelate without the need of counting on third people*. Using templates for these representations can also help improve the understandability of the artifacts. Moreover, counting on a major structure that holds and interrelates several software artifacts, would help reduce the effort to understand the product, the need to count on external people, and the loss of information produced by having the pieces distributed in different sources.

Concerning the participants

The regular roles played by the actors in multiprovider software evolution scenarios do not contribute to improving the conditions under which the next evolution project is going to be performed. On the one hand, the provider (mainly the project leader and the development team) has few or no major incentives to generate complete and up-to-date artifacts that help others evolve the software. In this scenario, the product ownership tends to rest in the provider, but such a role could change if the customer decides to continue the system evolution with a new provider, or the provider decides to no longer evolve such a product. These situations discourage providers from being the guardian of the product information.

On the other hand, the delivery of such information is usually not included in the project contract, and if it does, the client organization rarely has a guardian or a product owner who understands the artifacts and takes care of them. In fact, the participants in the study indicate that most customers are not aware of the relevance that software artifacts have in the product evolution; therefore, nobody pays extra money for them. In this sense, there is a challenge *to make customers and providers realize that the product artifacts are as valuable as the source code, and therefore, they need to be governed in a way similar to the source code*. If both parts understand it, then the software artifacts will become a marketable product that would help address most of the previous challenges.

VIII. Threats to validity

The results of this study should be understood in the context of small and medium-sized software enterprises (SMEs) that develop and maintain software in multiproviders scenarios. This exploratory study considered the construct, internal and external validity, as characterized in [Woh12].

To address the threats to the *construction validity*, we used semi-structured interviews as a data collection instrument that allowed us to collect qualitative and quantitative information. In order to eliminate possible ambiguities and biases in the questions, two external researchers evaluated the different versions of the inquiring instrument. Moreover, before applying the instrument we carried out a pilot interview, which allowed us to fine tune the questions, trying to achieve clear and direct answers.

Concerning the providers involved in this study, they represent a quite diverse set of SMEs, most of them were enrolled in a Chilean software providers community. The list of candidates that participate in the study was built based on the preselection performed by the SMEs, considering the inclusion and exclusion criteria. The participants were invited by email, where we explained to them the rights and conditions to participate in the interview and in the study. The information provided to the participants does not allow them to envision the elicitation or research questions, thus avoiding the bias of RQs guessing. In order to ensure accuracy of the PPOs answers when they refer to activities conducted by team members, all PPOs have played other technical roles as part of the development team during the last five years (inclusion criteria iv).

In order to mitigate the threats to the *internal validity* the processing and analysis of results followed the guidelines from Seaman [Sea99] and Eisenhardt [Eis89]. The participants gave their explicit consent to be recorded in video during the interview. The records allowed us to review the interviews more than once, and thus to recover trustworthy and contextualized information. Although the SMEs and participants were quite diverse, it was not possible to determine that the sample was representative of the target population. The sample was the result of applying the inclusion and exclusion criteria that ensured us to count on participants capable of providing valid information and opinions for this study. The answers of the participants showed no contradictory information, presenting similar or complementary information. This indicates that the reliability of the source information is probably high and representative of the sample.

Concerning the possible threats to *external validity*, this study recognizes the diversity of companies and contexts in which evolution projects are carried out, and clearly establishes the scenarios in which the reported results can be considered valid. Although the source information that supports the results is quite consistent, more empirical research is required to determine the eventual generalization of the findings. Given the dynamism of this activity, we recognize that the validity of the study findings will change as the artifacts and practices used to address PKAAs also change.

IX. Conclusions and Future Work

Performing the PKAAs is mandatory in software evolution projects. However, there is a diversity of evolution scenarios that involve different goals, people, software artifacts, and activities. In particular, the software evolution in multiprovider scenarios has received little attention from the software engineering community. The study reported in this article explores the way in which a set of SMEs from Chile and Brazil perform the PKAAs in two particular evolution scenarios.

The results indicate that the evolution process is mainly informal and artisanal, and the product ownership is temporal and tends to be a responsibility of the current provider. Usually, there is not a formal recognition for the provider to play such a role, which demotivates it to generate complete and reliable software artifacts, to make explicit the tacit knowledge, and to govern such a knowledge given its relevance for the next product evolution projects.

From the customer side, the process is not more auspicious, since it usually does not have a product owner (or someone who plays such a role). Typically, the customer is not aware of the relevance of the supporting information in the product evolution; therefore, it does not take care of it. Consequently, due to the lack of motivation of the provider or awareness of the client, much supporting information is lost after each evolution project. This loss becomes critical when the software provider changes. This situation keeps or worsens the initial conditions to be addressed by every new provider during the next evolution project.

Our main conclusion is that a paradigm change is required to break the status-quo in software evolution projects when they are performed in multiprovider scenarios. On the one hand, the software artifacts should improve its completeness, understandability and up-to-datedness. On the other hand, the artifacts should be governed and understood as a marketable component of the software, similar to the source code. In addition, the customer should count on a product owner able to understand, validate and take care of these artifacts.

These results illustrate the complexity and challenges for conducting these activities, the effectiveness of the product exploration process in each study scenario, and the main obstacles to improving that process. The study results also show several opportunities for research.

Acknowledgment

The research work of Anelis Pereira-Vale has been funded by ANID-Subdirección de Capital Humano/Doctorado Nacional/2021-21211216, Chile. We want to express our gratitude to those who contributed significantly in this study; thanks for your dedication and valuable comments.

References

[Alm23] Almashhadani, M., Mishra, A., Yazici, A., and Younas, M. “Challenges in Agile Software Maintenance for Local and Global Development: An Empirical Assessment”. *Information*, 14, 261, 2023. doi: https://doi.org/10.3390/info14050261.

[Bas90] Basili, V.R. “Viewing maintenance as reuse-oriented software development”. *IEEE Software*, June 47-55, 1990.

[Ben00] Bennett, K. and Rajlich, V. “Software maintenance and evolution: A roadmap”. In *Proceedings of Conference on the Future of Software Engineering (ICSE '00)*. Association for Computing Machinery, New York, NY, USA, pp. 73–87, 2000. doi: https://doi.org/10.1145/ 336512.336534.

[Ben02] Bennett, K., Rajlich, V. and Wilde, N. “Software Evolution and the Staged Model of the Software Lifecycle”. *Advances in Computers*, vol. 56, pp. 1-54, 2002.

[Ben08] Bennett C., Myers D., Storey M.A., Germán, D.M., Ouellet, D., Salois, M., and Charland, P. “A survey and evaluation of tool features for understanding reverse‐engineered sequence diagrams”. *Journal of Software Maintenance and Evolution: Research and Practice*, 20(4):291‐315, 2008.

[Ber13] Bergel, Maass, S., Ducasse, S., and Gîrba, T. “A Domain-Specific Language for Visualizing Software Dependencies as a Graph”. In *Proceedings VISSOFT IEEE Working Conference on Software Visualization*, pp. 45-49, 2014.

[Bib17] Biblioteca del Congreso Nacional de Chile. “Easy Law. Small and Medium-sized Enterprises Statute” (In Spanish: Ley fácil. Estatuto de las PYMES). 2017. Available at: https://www.bcn.cl/ portal/leyfacil/recurso/estatuto-de-las-pymes. Retrieved on June 2024.

[Bja15] Bjarnason, E., Unterkalmsteiner, M., Engström, E., and Borg, M. “An Industrial Case Study on Test Cases as Requirements”. In *Proceedings Agile Processes in Software Engineering and Extreme Programming XP*. Lecture Notes in Business Information Processing, vol 212. Springer, Cham, 2015, doi: https://doi.org/ 10.1007/978-3-319-18612-2_3.

[But22] Butt, S.A., Melisa, AC., and Misra, S. “Software Product Maintenance: A Case Study”. In *Proceedings Computer Information Systems and Industrial Management (CISIM)*. Lecture Notes in Computer Science, vol 13293. Springer, Cham, 2022. doi: https://doi.org/10.1007/978-3-031-10539-5_29.

[Cha11] Charrada, E.B., Caspar, D., Jeanneret C., and Glinz, M. “Towards a benchmark for traceability”. In *Proceedings of the 12th International Workshop on Principles of Software Evolution and the 7th Annual ERCIM Workshop on Software Evolution (IWPSE-EVOL)*. ACM, 2011:21-30.

[Cor09] Cornelissen B., Zaidman A., Van Deursen, A., Moonen, L., and Koschke, R. “A systematic survey of program comprehension through dynamic analysis”. *IEEE Trans Softw Eng*. 35(5):684‑702, 2009.

[Eis89] Eisenhardt, K. M. “Building theories from case study research”. *Academy of Management Review* 14, pp. 532-550, 1989.

[Eis01] Eisenbarth, T., Koschke, R., and Simon, D. “Aiding program comprehension by static and dynamic feature analysis”. In *Proceedings of the IEEE International Conference on Software Maintenance*, IEEE; pp. 602‑611, 2001.

[Feb16] Febrero, F., Calero, C., and Moraga, M. A. “Software reliability modeling based on ISO/IEC SQuaRE”. *Information and Software Technology*, 70, pp. 18–29, 2016.

[Fra23] Frattini, J., Fucci, D., Mendez, M., Spínola, R., Mandić, V., Taušan, N., Ahmad, M. O., and Gonzalez-Huerta, J. “An initial theory to understand and manage requirements engineering debt in practice”. *Information and Software Technology*, 159, 107201, 2023.

[Fri12] Fricker, S.A. “Software Product Management”. *Management for Professionals in Software for People*, edition 127, pp. 53-81, Springer. 2012.

[Ful08] Shull, F., Singer, J., and Sjøberg, D.I.K. “Guide to advanced empirical software engineering”. *Springer Science & Business Media*, 2008. doi: https://doi.org/10.1007/978-1-84800-044-5.

[Gar09] Garlan, D., Barnes, J.M., Schmerl, B., and Celiku, O. “Evolution Styles: Foundations and Tool Support for Software Architecture Evolution”. In *Proceedings of Working IEEE/IFIP Conference on Software Architecture (WICSA)*, 2009. doi: https://doi.org/10.1109/WICSA.2009.5290799.

[Gha18] Ghaleb, T.A., AlTurki, M. A., and Aljasser, K. “Program comprehension through reverse-engineered sequence diagrams: A systematic review”. *Journal of Software: Evolution and Process*, 30(11), 2018. doi: https://doi.org/10.1002/smr.1965.

[Gvr23] Grand View Research. “Custom Software Development Market Size, Share, & Trends Analysis Report By Deployment (Cloud, On-premise), By End-use (BFSI, Government), By Enterprise Size (SMEs, Large), By Solution, By Region, And Segment Forecasts, 2023 - 2030”. *Report ID: GVR-4-668040-001-6*, pp. 120, 2023. Summary available at: https://www.grandviewresearch.com/industry-analysis/custom-software-deve-lopment-market-report. Retrieved on June 2024.

[Hal08] Hall, T., Beecham, S., Verner, J., and Wilson, D. “The impact of staff turnover on software projects: the importance of understanding what makes software practitioners tick”. In *Proceedings of the ACM SIGMIS CPR conference on Computer personnel doctoral consortium and research (SIGMIS CPR '08)*. Association for Computing Machinery, New York, NY, USA, 30–39, 2008, doi: https://doi.org/10.1145/ 1355238.1355245.

[Her13] Herraiz, I., Rodriguez, D., Robles, G. and Gonzalez-Barahona, J.M. “The evolution of the laws of software evolution: A discussion based on a systematic literature review”. *ACM Comput. Surv*. 46(2), Article 28, 2013.

[ISO17] ISO/IEC/IEEE International Standard. "ISO/IEC/IEEE International Standard -Systems and software engineering- Software life cycle processes". In *ISO/IEC/IEEE 12207:2017(E) First edition 2017-11*, pp.1-157, 15 Nov. 2017. doi: https://doi.org/10.1109/IEEESTD.2017.8100771.

[ISO22] ISO/IEC/IEEE International Standard. "ISO/IEC/IEEE International Standard -Software engineering - Software life cycle processes - Maintenance". In *ISO/IEC/IEEE 14764:2022(E)*, pp.1-46, 21 Jan. 2022, doi: https://doi.org/10.1109/IEEESTD.2022.9690131.

[Jab13] Jaber. K., Sharif. B., and Liu C. “A study on the effect of traceability links in software maintenance”. *IEEE Access*, 1:726-741, 2013.

[Jef19] Jeffery, C.L. “The city metaphor in software visualization”. In *Proceedings of the 27th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG 2019)*, 2019. url: http://hdl.handle.net /11025/35620.

[Kbv22] KBV Research. “Global Custom Software Development Market Size, Share & Industry Trends Analysis Report By Solution, By End User, By Enterprise Size (Large Enterprises and Small & Medium Enterprises), By Deployment (Cloud and On-premise), By Regional Outlook and Forecast, 2022 - 2028”. *Report ID: KBV-12428*. November 2022. Summary available at: https://www.kbvresearch.com/custom-software-development-market/. Retrieved in June 2024.

[Kra22] Krause-Glau, A., Bader, M., and Hasselbring, M. “Collaborative Software Visualization for Program Comprehension”. In *Proceedings of Working Conference on Software Visualization (VISSOFT)*, pp. 75-86, Limassol, Cyprus, 2022. doi:https://doi.org/10.1109/VISSOFT55257.2022.00016.

[Kru09] Kruchten, P., Capilla, R., and Dueas, J. “The Decision View's Role in Software Architecture Practice”. *IEEE Software*, 26(2), 36-42, 2009.

[Lav11] Lavallée, M., and Robillard, P.N. “Causes of premature aging during software development: an observational study”. In *Proceedings of the 12th International Workshop on Principles of Software Evolution and the 7th annual ERCIM Workshop on Software Evolution (IWPSE-EVOL '11)*. Association for Computing Machinery, New York, NY, USA, 61–70, 2009.

[Len17] Lenarduzzi, V., Sillitti, A., and Taibi, D. “Analyzing Forty Years of Software Maintenance Models”. In *Proceedings 39th International Conference on Software Engineering Companion (ICSE-C)*, IEEE/ACM Buenos Aires, Argentina, pp. 146-148, 2017. doi: https://doi.org/10.1109/ICSE-C.2017.122.

[Lap18] Lappi, T., Karvonen, T., Lwakatare, L.E., Aaltonen, K., and Kuvaja, P. “Toward an Improved Understanding of Agile Project Governance: A Systematic Literature Review”. *Project Management Journal*, 49(6): 39-63, 2018, doi: https://doi.org/10.1177/8756972818803482.

[Mad15] Mäder, P., and Egyed, A. “Do developers benefit from requirements traceability when evolving and maintaining a software system?”. *Empir Softw Eng.*, 20(2): 413-441, 2015.

[Mat06] Mattsson, M., Grahn, H., and Martensson, F. “Software architecture evaluation methods for performance, maintainability, testability, and portability”. In *2nd International Conference on the Quality of Software Architectures*, 2006.

[Mat16] Mattila, A.L., Ihantola, P., Kilamo, T., Luoto, A., Nurminen, M., and Väätäjä, H. “Software visualization today: systematic literature review”. In P*roceedings of the 20th International Academic Mindtrek Conference (Academic Mindtrek'16)*. ACM Press, New York, NY, USA, 262–271, 2016. doi: https://doi.org/10.1145/2994310.2994327.

[Men05] Mens, T., Wermelinger, M., Ducasse, S., Demeyer, S., Hirschfeld, R. and Jazayeri, M. “Challenges in software evolution”. In *Proceedings Eighth International Workshop on Principles of Software Evolution (IWPSE'05)*, Lisbon, Portugal, 2, pp. 13-22, 2005. doi: https://doi.org/10.1109/IWPSE.2005.7.

[Min14] Ministerio de Economía, Fomento y Turismo. Gobierno de Chile. “Background for the Review of the Classification Criteria of the Small and Medium-sized Enterprises Statute” (In Spanish: Antecedentes para la revisión de los criterios de clasificación del Estatuto PYME), 2014. Available at: https://www.economia.gob.cl/wp-content/uploads/2014/04/Boletin-Revision-Clasificacion-Estatuto-Pyme.pdf. Retrieved on June 2024.

[Muh22] Muhammad, A.P., Knauss, E., Batsaikhan, O., Haskouri, N.E., Lin, YC., and Knauss, A. “Defining Requirements Strategies in Agile: A Design Science Research Study”. In *Proceedings of the International Conferences on Product-Focused Software Process Improvement (PROFES’22)*. Lecture Notes in Computer Science, vol. 13709. Springer, Cham, 2022, doi: https://doi.org/10.1007/978-3-031- 21388-5_6.

[Ozk10] Ozkaya, I., Wallin, P., and Axelsson, J. “Architecture knowledge management during system evolution: observations from practitioners”. In *Proceedings of the 2010 ICSE Workshop on Sharing and Reusing Architectural Knowledge (SHARK '10)*. ACM Press, New York, NY, USA, pp. 52–59, 2010.

[Pfa20] Pfahler, F., Minelli, R., Nagy, C., and Lanza, M. “Visualizing evolving software cities”. In *Proceedings of the IEEE Working Conference on Software Visualization (VISSOFT’20)*, pp. 22–26, 2020. doi: https://doi.org/10.1109/VISSOFT51673.2020.00007.

[Pin05] Pinelle, D., and Gutwin, C. “A Groupware Design Framework for Loosely Coupled Workgroups”. In *Proceedings of the 9th European Conference on Computer-Supported Cooperative Work (ECSCW’05)*, Springer, Dordrecht. Paris, France, 18-22 September 2005.

[Raj00] Rajlich, V., and Bennett, K.H.“A staged model for the software life cycle”. *IEEE Computer*, 33, 66–71, 2000.

[Ras19] Rashid, M., Clarke, P.M., and O’Connor, R.V. “A systematic examination of knowledge loss in open source software projects”. *International Journal of Information Management*, 46, pp. 104-123, ISSN 0268-4012, 2019. doi: https://doi.org/10.1016/ j.ijinfomgt.2018.11.015.

[Rob21] Robillard, M.P. “Turnover-induced knowledge loss in practice”. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2021)*, 2021. doi: https://doi.org/10.1145/3468 264.3473923.

[Sal21] Salama, M., Bahsoon, R. and Lago, P. “Stability in Software Engineering: Survey of the State-of-the-Art and Research Directions”. *In IEEE Transactions on Software Engineering*, 47(7): 1468-1510, 1 July 2021.

[San13] Sandoval, J.P., Bergel, A., Ducasse, S., and Denker, M. “Performance evolution blueprint: Understanding the impact of software evolution on performance” . In *Proceedings of the IEEE Working Conference on Software Visualization (VISSOFT)*, 1-9, 2013.

[Sav15a] Savolainen, P., Ahonen, J.J., and Richardson, I. “ When Did Your Project Start? - The Software Supplier’s Perspective”. *Journal of Systems and Software*, 104, C, 32-40. June 2015.

[Sav15b] Savolainen, P. and Ahonen, J. J. “Knowledge lost: Challenges in changing project manager between sales and implementation in software projects”. *International Journal of Project Management*, 33(1): 92-102. 2015.

[Sch10] Schrettner, L., Hegedũs, P., Ferenc, R., Fülöp, L. J, and Bakota, T. “Development of a Methodology, Software Suite and Service for Supporting Software Architecture Reconstruction”. In Proceedings *14th European Conference on Software Maintenance and Reengineering*, pp. 190-193, Madrid, Spain, 2010. doi: https://doi.org/10.1109/CSMR.2010.32.

[Sea99] Seaman, C.B. “Qualitative methods in empirical studies of software engineering”. *IEEE Trans. Softw. Eng*. 25: 557-572, 1999.

[Smo17] Smolander, S. “High performance work practices and project manager turnover in IT projects”. *Master’s Thesis in Computer Science in University of Vaasa*, 2017.

[Sne05] Sneed, H.M., Hasitschka, M., and Teichmann, M.T. “Software product management: Maintenance and further development of existing application systems” (In German: Software-Produktmanagement: Wartung und Weiterentwicklung bestehender Anwendungs-systeme), *dpunkt.verlag*, 2005.

[Sto21] Stojanov, Z. “Software maintenance improvement in small software companies: Reflections on experiences”. In *Proceedings of the 3rd International Workshop on Information, Computation, and Control Systems for Distributed Environments (ICCS-DE)*. pp. 182-197, 2021. Irkutsk, Rusia.

[Tia21] Tian, F., Wang, T., Liang, P., Wang, C., Khan, A. A., and Babar, M.A. “The impact of traceability on software maintenance and evolution: A mapping study”. *Journal of Software: Evolution and Process*, 33(10), 2021. doi: https://doi.org/10.1002/smr.2374.

[Tri15] Tripathy, P., and Naik, K. “Software Evolution and Maintenance: A Practitioner’s Approach”. *John Wiley & Sons*: Hoboken, NJ, USA, 2014.

[Wet07] Wettel, R. and Lanza, M. “Visualizing software systems as cities”. In *Proceedings of the 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis*, pp. 92–99, 2007. doi: https://doi.org/10.1109/VISSOF.2007.4290706.

[Woh12] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., and Wesslén, A. “Experimentation in Software Engineering”. *Springer Berlin, Heidelberg*, 2012. doi: https://doi.org/10.1007/978-3-642-29044-2.