2013 17th European Conference on Software Maintenance and Reengineering

An Exploratory Study of Cloning in Industrial Software Product Lines

Yael Dubinsky*, Julia Rubin*†, Thorsten Berger‡§, Slawomir Duszynski¶, Martin Becker¶ and Krzysztof Czarnecki‖
*IBM Research in Haifa, Israel. Email: {dubinsky,mjulia}@il.ibm.com
†University of Toronto, Canada.
‡IT University of Copenhagen, Denmark. Email: thbe@itu.dk
§University of Leipzig, Germany.
¶Fraunhofer IESE, Germany. Email: {slawomir.duszynski,martin.becker}@iese.fraunhofer.de
‖University of Waterloo, Canada. Email: kczarnec@gsd.uwaterloo.ca

***Abstract*—Many companies develop software product lines—collections of similar products—by cloning and adapting artifacts of existing product variants. Transforming such cloned product variants into a "single-copy" software product line representation is considered an important software re-engineering activity, as reflected in numerous tools and methodologies available. However, development practices of companies that use cloning to implement product lines have not been systematically studied. This lack of empirical knowledge threatens the validity and applicability of approaches supporting the transformation, and impedes adoption of advanced solutions for systematic software reuse. It also hinders the attempts to improve the solutions themselves.**

We address this gap with an empirical study conducted to investigate the cloning culture in six industrial software product lines realized via code cloning. Our study investigates the processes, and the perceived advantages and disadvantages of the approach. We observe that cloning, while widely discouraged in literature, is still perceived as a favorable and natural reuse approach by the majority of practitioners in the studied companies. This is mainly due to its benefits such as simplicity, availability and independence of developers. Based on our observations, we outline issues preventing the adoption of systematic software reuse approaches, and identify future research directions.

***Keywords*-software product line; cloned product variants; exploratory study;**

I. Introduction

Software Product Line Engineering (SPLE) approaches support development of products from a common set of core assets in a prescribed way [1], [2], [3]. These approaches advocate strategic, planned reuse that yields predictable results. However, in reality, software product lines often emerge *ad-hoc*, when a company has to address its target market needs by releasing a new product that is similar, yet not identical, to existing ones. In many cases, artifacts of an existing product are *cloned* and *modified* to fit the new requirements—the "clone-and-own" approach [4], [5], [6].

Usage of the "clone-and-own" approach is discouraged in the SPLE literature and numerous authors propose solutions for re-engineering cloned products into a "single-copy" software product line representation [4], [5], [7], [8], [9], [10], [11]. Why is cloning then still a popular method of choice in many industrial organizations?

The first step towards suggesting alternatives that can replace cloning is to gain a better understanding of the current cloning practices. Are the organizations satisfied with those practices? What are the advantages and disadvantages of the approach? Do they wish to eliminate cloning? And if so, what prevents them from doing that?

Surprisingly, to this date, no systematic study has been conducted to investigate these questions, and little empirical knowledge is available. Without such a study or empirical knowledge, understanding and improving the current cloning practice remains difficult and unfocused.

We thus conducted an exploratory study to investigate the cloning culture in the context of software product lines. Specifically, we performed eleven semi-structured interviews with participants involved in six cloned product lines of three well-established large-scale organizations providing solutions in the data storage, aerospace and defense, and automotive domain. We interviewed employees from a variety of roles, such as developers, testers and product managers. We used a structured questionnaire to complement the interview data and analyzed all the collected data using the grounded theory approach [12].

Among our findings, we observed that the majority of product lines that use cloning techniques still perceive it as a favorable reuse approach, mainly because it is a *rapidly* available mechanism that allows practitioners to start from an already *specified and verified* functionality, while having the *freedom and independence* to make any necessary modifications. In fact, several practitioners are satisfied with the cloning practices and believe that it is a viable reuse mechanism. While others would wish to shift to a better managed approach, we observed that existing organizational structures might impede the shift because there is usually no role in an organization that is responsible for promoting reuse.

Based on these findings, we develop a set of recommendations for practice, and suggestions for future research. We address organizational structure, governance models, and tool support that are required to efficiently develop and manage software product line assets. With our work, we aim to assist both practitioners and researchers who are interested in the field of software reuse: practitioners can use the experience gained by other organizations and leverage it in their own settings, while researchers and tool developers can acquire better understanding on why and how cloning is practiced in the industry and refine their proposed solutions accordingly.

We proceed as follows. Section II introduces our research methodology. Section III reports the main results, followed by a discussion on their implications in Section IV. Section V depicts limitations and threats to validity, and Section VI summarizes related work. Finally, Section VII concludes.

II. Research Methodology

We base our exploratory study on *grounded theory*—a methodology developed by Glaser and Strauss [12] for the purpose of building *theory* from data. It was later refined and generalized by Corbin and Strauss [13] to derive *theoretical constructs* from qualitative analysis. Grounded theory is increasingly being used in studies related to Software Engineering [14], because of its ability to formulate a theory based on empirical evidence.

The objective of our study was to understand how cloning happens in the observed organizations and what its underlying mechanisms are, rather than confirming or refuting any specific hypotheses. Thus, this work has an exploratory, theory-building nature. We followed the approach in [13], which recommends evolving the research tools in the course of the research, as more information becomes available to take advantage of the emerging results. For example, the questionnaire that was used in the first interview was later reduced to a shorter attitude questionnaire. In addition, after the fourth interview, we decided to interview several participants from a specific product line. This was done to gain an extended perspective on the product line governance issues, involving a number of members from the same team.

A. Participants

As summarized in Table I, we interviewed eleven practitioners from six different product line (PL) environments in three different enterprise organizations that belong to three different industries: aerospace and defense (A&D), data storage management (DSM), and automotive (Auto). The selection of the organizations was limited to those we had access to and that use cloning as a method to reuse artifacts between the different products in their product lines.

The first product line we analyzed in our study (PL1) is relatively new: it emerged around three years ago. The second and the third ones (PL2 and PL3) are mature product lines developed for eight to ten years each. Products of PL4 were initialized one and a half year ago as proofs of concept, and were later commercialized as they became successful. PL5 was developed over the course of five years and nowadays is being merged with a larger product line, due to a company acquisition. The first products of PL6 were created about 10 years ago, and further products emerged continually up to the present time.

Table I
Interview participants (ordered chronologically)

Participant	Industry	PL	Role
p1	A&D	1	Software leader
p2	A&D	2	Senior software leader
p3	A&D	3	Integrator & QA engineer
p4	A&D	4	Architect
p5	A&D	2	Developer
p6	DSM	5	Architect
p7	DSM	5	Technical leader
p8	DSM	5	Senior architect
p9	DSM	5	Developer
p10	DSM	5	Technical leader
p11	Auto	6	Technical leader & developer

Almost all ever created products of these product lines are still sold on the market. Both full-product cloning and feature cloning are used in the interviewed organizations. Development teams mostly reuse artifacts developed “in house”—by the developers of the same team.

From 26 to 100 people are involved in developing and managing each product line. To gain a broad perspective, we selected representative roles that have decision rights and responsibilities at various stages of the product line development (column “Role” in Table I). The theoretical sampling was evolved when after the fourth interview, we decided to extend our point of view by going back and interviewing another participant from product line #2 (participant p5). Also, in order to gain team perspective we interviewed five participants from the same product line (participants p6–p10). The last interview (participant p11) was already scheduled before that decision, thus we were only able to reach one person from that organization.

B. Data Collection

We used a semi-structured interview and a pre-interview attitude questionnaire [15] to collect our data. Combining both research tools provided us with the ability to triangulate data when identifying the concepts and categories, as well as to enrich our description of the results.

Questionnaire. The interviewees were requested to spend between five to ten minutes on filling in the questionnaire right before starting the interview. Its goal was to gather interviewees’ initial perception, before starting the conversation. When filling in the questionnaire, participants were asked to think about the product line they work on and to indicate the extent to which each of the questionnaire statements describes it. Together with statements about the cloning practices, the questionnaire included statements that relate to the product line setting in general.

Table II presents an aggregated view of answers to some of the questionnaire’s statements. We only show statements that strongly support or contradict the derived theory (Section III)

Table II
RESULTS OF THE PRE-INTERVIEW ATTITUDE QUESTIONNAIRE, USED TO ELICIT PARTICIPANTS' AGREEMENT WITH CERTAIN STATEMENTS. THE TABLE SHOWS THE NUMBER OF PARTICIPANTS CHECKING EACH CELL.

	Statement	Not at all				To a great extent
1.	I'm aware of a product line strategy in my organization	1		2	2	6
2.	My team develops core assets that are later used by other teams	1	2	4	3	1
6.	We regularly clone pieces of code		3	5	2	1
10.	We measure how many times a certain core asset is used	5	4		2	
17.	People in my team know who should approve a change request			1	4	5
19.	We have work procedures that include cloning of artifacts	2	6	1	2	
20.	I feel our development process is well-defined		1	2	6	2
21.	We clone to reuse artifacts between products of a product line			5	5	1
22.	All project teams follow in practice the defined process	1		2	6	2
23.	We have relatively many clones of code		6	4	1	
27.	We do not change main APIs without receiving an approval			1	5	5
31.	We have relatively big clones of code		5	4	2	

and are relevant to our discussion (Section IV). The number in each cell indicates the number of participants who marked this cell. For example, six participants indicated that they extensively clone to reuse artifacts between products of a product line (two rightmost columns in statement #21).

Interview Following the questionnaire, we conducted interviews that included open-ended questions and took about one hour. Each interview started with general questions regarding the product line environment and the interviewee role, and then focused on the way the cloning is performed. For example, we asked the interviewee to describe the product line that he or she is involved with, its lifecycle and phases, and the tools that are used.

Specifically, we asked about the capabilities of the tools, as well as about the roles and activities that are directly related to the product line development. We investigated the way cloning is implemented, e.g., who decides to make a clone and how the information about existing clones is maintained. We were interested to find out whether there are procedures that regulate the cloning process and whether there are any measurements in this respect. We also asked participants to describe which artifacts are cloned and at which phases, how big and how many clones there are.

C. Data Analysis

We used *open coding* [13] to analyze the collected data. The transcripts of our interviews and the answers to the questionnaire were analyzed line by line to detect *concepts*—key ideas contained in data. When looking for concepts, we searched for the best word or two that describes conceptually what we believe is indicated by the raw data. Based on the concepts, we created more abstract *categories* that group lower-level concepts and represent threads of ideas that we found in the analyzed data. We then related concepts to each other, forming a hierarchy of concepts and sub-concepts. This hierarchy is presented in Section III.

In grounded theory, data collection and analysis are inter-related processes: we used evolving concepts and categories to refine questions that we ask subsequent participants. This process is referred to as *theoretical sampling* and was illustrated in Section II-A. The process continued until no new concept emerged—thus, *conceptual saturation* is reached. Since all findings are linked to specific evidence—either an interview quote or a questionnaire response—we are able to provide grounding to the results we report and attribute them to the collected data.

III. RESULTS: CLONING PRACTICES

This section presents the findings of our study. Following the analysis of the interviews and the questionnaires, we learned that cloning happens at all stages of software development—from requirements through design, implementation and testing, up to the preparation of user documentation: artifacts in each stage are cloned and adjusted to the needs of a new customer. Practitioners, however, indicated that in their projects, code is cloned more often than other artifacts.

All but three practitioners indicated that, by design, their products are built on top of a reusable platform. Still, cloning is a frequent reuse mechanism and commonalities are rarely extracted to the common platform. A team lead of a product indicated that:

"there are [core, reusable] assets but they are degenerated, just place-holders; < ... > [core asset] repository does not contain significant assets, but the intention is there."

We also observed that clones can be of different size, from few methods or single components up to significant sub-products and even complete products. The amount of clones in a product line varied between a few (five to ten) instances up to few tens of instances.

Below, we identify and summarize four major characteristics (categories) related to the use of cloning in software product lines context, which are further discussed in Sections III-A–III-D. When presenting our empirical observations, we do not attempt to assess their positive and negative implications; these implications are discussed in Section IV.

Efficiency: Cloning is perceived to be a simple yet efficient reuse mechanism that saves time and resources. It allows participants to start the development from already implemented and verified set of artifacts. At the same time, it provides independence and freedom to change these artifacts as needed.

Overhead: Adapting cloned artifacts to the new needs sometimes involves a significant effort. Effort in maintaining the artifacts can also be increased because some tasks need to be performed on each cloned copy. Also, propagating modifications between clones is not a trivial task.

Short-Term Thinking: Lack of resources to invest in systematically managing reuse, as well as lack of awareness of other reuse approaches, leads to choosing cloning as the favorite reuse mechanism. Organizations often focus on making sure their individual products are successful and postpone dealing with reuse issues to the future.

(Lack of) Governance: Knowledge about reuse is rarely maintained. Reuse is not measured and there are usually no roles in the organization that are responsible for reuse practices and processes.

In what follows, we discuss these observations in more details and illustrate them with the quotes from the participants.

A. *Efficiency*

There are three major reasons for considering cloning as an efficient reuse mechanism in software product line engineering.

1. Cloning saves time and reduces costs: Most interviewees stated that it is faster and more efficient to start with an already developed and tested set of artifacts:

"It is easier to start with something. Cloning gives [us] an initial basis."

"< ... > there is no need to reinvent the wheel."

"The most significant thing < ... > is to take a 'stack' as is and reuse it. This can save thousands of hours."

"We want to make the development faster. < ... > The work assumption is that we clone."

Moreover, most participants highlighted the importance of reusing the already *validated* code:

"It saves time. These components were already used, tested, closed. A kind of an on-the-shelf software."

"We did something. It is 'old' and for most cases it is stable. The amount of time to bring [new code] to the required level of quality is not easily estimated."

2. Cloning provides independence: After cloning, developers enjoy the freedom of making any necessary changes to their clones. They do not have to worry about synchronization with other teams, neither about the form of the common artifacts nor about their lifecycle and schedule.

"It gives freedom to change, [when cloning] there is no damage to existing products."

"The management is more convenient. We believed that this code will be significantly changed so no point in keeping it as a shared code."

"No code sharing, [because] the two copies are not developed in the same pace."

Also, one participant emphasized the understandability and readability of the cloned product-specific code compared to the more complex, generic reusable code:

"[In the past,] a new variant < ... > was integrated back into the mainstream by using preprocessor switches. This has made the code very unreadable, so we wanted to go away from that and we started to branch off the files that differ among variants."

3. Cloning is the most available mechanism: Some of our participants indicated that there are no other readily available reuse mechanisms:

"There are simply no other options."

"There was no other choice. [But] we need to avoid doing cloning, it has a price."

B. *Overhead*

As one of the interviewees noted in the previous section, cloning practices have their price: organizations usually have to deal with the overhead involved in creating and managing the clones. We identified four major issues introduced by cloning.

1. Propagating changes between clones is difficult: Most interviews stated that no connection between clones is maintained. Thus, it is difficult to make sure that changes and bug fixes made to one of the clones are propagated to the others.

"< ... > code that we cloned looses connection with the product which it is cloned from, and then there is no sharing of new insights and innovations."

"If we find a bug then many times it can be here and also in other places. The new product contains code that exists also in the old product. So, if we fix the old one then we also fix the new or vice versa."

"Sometimes, we find the same bug again in a different variant that nobody thought about before."

2. Integration of the cloned artifacts is difficult: In some cases, practitioners stated that adapting clones to the new needs involves larger effort than expected.

"In this process we always lose quality. Sometimes, we have no choice but to throw away the code and re-write."

"It is usually not possible to port without making changes to the code."

"[They] took an existing asset and tried to reuse. They claimed that integration duration was too long."

"Sometimes clones are too big for relatively small needs."

"It is a copy and a lot of adaptation."

3. Repetitive tasks are common: Some participants indicated that cloning causes a significant increase in the maintenance effort.

"We need to perform many activities several times: for each variant, we have to check the code and implement the change or fix. Then, the design and documentation documents, as well as the test specification need to be adapted for each variant. Tests need to be run."

4. It is not clear which variant to use as the source for cloning: One participant indicated that when several clones exist, it is not always clear which one is the best "starting point" for cloning.

"With each new project we ask: 'where do you start from'?"

C. Short-Term Thinking

As stated in Section III-A, sometimes, practitioners perceive cloning as the only available reuse mechanism. This often occurs because development organizations invest little time and resources in supporting and managing reuse. Below are three major factors reflecting organizational thinking with respect to reuse.

1. Lack of planning: Most practitioners stated that thinking ahead and planning for reuse is rare.

"We were not aware we develop product lines."

"Maybe we can [think about reuse] from the beginning. Still this is easy to say now, when we know that the first product is a success. At the beginning, the other risks are more important."

"When a new customer came, we needed to decide how to implement his requirements in the fastest way. We do not have time to think thoroughly about generic approaches."

On the other hand, sometimes it is not clear in the beginning of product development that reuse would be needed:

"At the beginning we did not know that we will have to support all the controllers that we support now – this emerged over time."

2. Lack of resources: Our interviewees were concerned with the lack of support for product line engineering practices.

"There is a lack in resources for an organized work and methodology with respect to the product line engineering."

"We sometimes need to beg for reuse."

3. Unawareness of other approaches: In some cases, cloning is considered to be "state-of-the-art" in reuse.

"We clone code and should do better with cloning requirements and design."

"If something is good $< \dots >$ then it will be cloned."

Only in one case, the participant explicitly stated that even though other approaches were considered, the company deliberately chose cloning as their reuse approach:

"We explicitly decided to use separate branches for our variants."

D. (Lack of) Governance

All but one of the participants clearly stated that no governance of product line development exists in their organizations. We identified three issues related to the (lack of) product line governance.

1. Lack of reuse tracking: All practitioners indicated that no infrastructure that tracks and facilitates reuse opportunities exists. Information about cloned artifacts exists primarily in people's mind and they are responsible to make sure that changes between clones are propagated correctly. Similarly, reuse opportunities are identified if somebody *remembers* which similar artifact can be reused. Reuse is done via personal knowledge, memory and networking.

"When requirements are given, the software leader and most of us know if there is already such a thing."

"I am aware of things that I did and saw. If I recall something similar to what I need, I'll find it and copy."

"A person who was in a specific team takes the capabilities to another project."

"Many things are in the heads of people: 'why don't you use what we did?"'

2. Lack of organizational roles and processes: In most cases, there are no organizational roles that are responsible for reuse. Many times, project leads are those who actively look for reuse opportunities, in order to reduce costs of their tasks. However, there were no participants who indicated that they are *encouraged* to contribute reusable artifacts.

"No one [is responsible for reuse]. One who requires an asset, takes it."

"No one is in charge of the cloning knowledge—in practice, it is the one who implements [a functionality] and the architect who is in charge of the work item."

"The decision to do cloning is probably done by a manager or an architect."

"The responsibility at the end is on the software leaders."

"In each project, there is a software leader who manages the software development activities as part of the system

development. He or she is in charge of using existing assets."

Only in two cases, the interviewees indicated that there is an organizational or technical structure that mandates the process of reuse.

"There is an architecture forum that is being led now by the person who is in charge of the assets management. There are members of all the disciplines in the department in this forum. The forum's role is to manage the department assets and identify artifacts that can be reused."

"We have a pool of components or files that are meant to be reused by the projects (using branching). However there is no dedicated group that maintains these assets; this is the responsibility of the projects."

In another case, a participant stated that there is a person who can *technically* clone artifacts when asked to.

"Developers are not supposed to clone. It happened at the beginning, but not later. The configuration manager has a procedure how to do it."

However, similarly to the lack of roles, there are mostly no processes that define when and how to clone artifacts.

"There is no place or procedure that asks to search for existing assets."

"It is not perceived as a process. It is simply something that is done."

3. Lack of measurement: In none of the cases we observed measurement of reuse. Organizations lack any quantitative indication on the benefits or drawbacks related to cloning.

"In quarterly reviews we should report how many hours in average we saved by reusing. [But there are] no measures. Usually there are reports on few hundreds of hours."

"There is no structured method [to evaluate reuse]. [We do] design review in which we validate that reuse was done."

"We don't really measure, but there are some places that know the level of reuse, e.g., that we use an asset four times. In general, no one measures."

IV. Discussion

In this section, we discuss the findings and the observations we made during data collection and analysis. We derive recommendations for practitioners, and identify directions for further research. Our discourse uses three different angles:

1) The human individual perspective that looks at the attitudes and motivation for cloning.
2) The technical perspective that deals with the way cloning is implemented and how it is merged with the other development activities.
3) The organizational perspective, where we examine the process descriptions and the management style that position the cloning practice as one of the practices that are used. We also examine the mechanisms that monitor the cloning enactment.

A. The Individual Perspective

Among our subjects, cloning is perceived as a natural technique to support the development of similar products. This is due to its low entrance barrier, being an easy thing to do since artifacts are ready and available. Also, in most cases, practitioners can perform cloning in an ad-hoc manner without the need to adhere to formal procedures. Most practitioners are happy with cloning and can easily explain their motivation, e.g.: it accelerates development, since we use what we already have; it saves time and, therefore, it saves money.

However, there is ambivalence in the way cloning is perceived. Delving into the details and raising questions on how cloning is performed and how clones are evolved and maintained reveals feelings such as frustration and helplessness as well as statements calling to avoid cloning or to "eliminate it". The cloning information is usually kept by each individual and no special tools are used to store, maintain, and share it. Maintaining different products that include independently-evolved cloned artifacts is time-consuming and gives the interviewees the feeling that "we are not professional enough".

In light of the ambivalence that was found, we suggest to further explore the cloning practice. Since it has clear advantages i.e., simplicity and availability, we suggest to study ways for resolving the causes of frustration and to define the terms in which cloning can exist and be promoted. We also suggest to compare between product line environments that use cloning and those that avoid cloning, thus improving the understanding of the alternatives. Any approach that aspires to be better than cloning has to have a way to address the great perceived advantages of cloning, which is simplicity and availability. Many approaches fail because they fail to convince practitioners that they would yield better results. Hard data would play a role in providing correct guidance and improving adoption of such alternative techniques.

In addition, it seems that practitioners lack the awareness and knowledge about different forms of reuse and, specifically, about the methods that avoid cloning. Promoting education on this topic can contribute to improving practices of individuals and organizations. Investing in education includes, among others, studying and leveraging experiences of other organizations. For example, our interviewees indicated that reusing already-tested code is one of the main benefits of cloning—while other companies experienced [16] that the original testing is not necessarily sufficient for the target conditions of the cloned code[1].

[1]A prominent case is the Ariane disaster [17], which occurred due to reuse of code that was tested and verified on previous missions, but which did not fit the new advanced system.

B. The Technical Perspective

As shown in Table II, code is regularly cloned (statement #6). Still, the level of reuse across the product line is not always perceived as high (statement #2) or just not measured (statement #10). Although the work procedures are defined and participants comply (statements #17, 20, 22, 27), they do not include the cloning activity (statement #19), which means that the cloning practices are performed in an ad-hoc manner.

One way to improve the cloning practice is to manage the cloning knowledge. Among others, interviewees suggested to document cloning tasks, meaning both to *identify* a task as a cloning one and to *document* its lifecycle. Such documentation can provide the cloning history, the ability to share the cloning knowledge and to examine the cloning patterns. While there are numerous tools that detect software clones, suggest refactoring techniques for their elimination, or promote actions to keep the clones synchronized (see Section VI), it is not clear at what level to track clones, which clones to track, and how to do it in the most effective way. To identify precise requirements for clone management tools in SPLE, or to determine suitable existing tools, is still a question for future studies. Moreover, one needs to also understand when it pays off to eliminate clones and when it may be better to keep them. Unfortunately, the SPLE community has little or no guidance on that.

Some practitioners also suggested strengthening the cloning practice by integrating it into the architectural decision making process. This means that cloning activities should be performed upon approval, should be consistent with the architectural decisions and should be validated during the architecture and design reviews. This would increase the transparency of the cloning knowledge and enable using this knowledge for better reuse.

Finally, there is lack of quantitative data on how much clones save or cost under different circumstances and how far we can scale cloning, i.e., for how many products in a product line cloning is worthwhile. Studies that quantify these issues are required. Also, there are trade-offs based on different risks whether to clone or to share [16]. There is a need for a better guidance on when and what to share versus clone.

C. The Organizational Perspective

Cloning, if applied, is part of the product line strategy that is used—whether it is explicitly defined or can be implicitly understood from the procedures that are derived from the strategy. We found that usually cloning is deployed using indirect ways. One of the participants stated:

> *"It would be good if there was a cloning process that is arranged and [there were] documents that define this practice."*

As we found out, processes that regulate reuse are missing in the interviewed companies and the organizational roles that are responsible for reuse are not well defined. Measurements regarding cloning are usually not taken (statement #10 in Table II) and there is a lack of tools to support the cloning process. However, as shown in Table II, eight out of eleven participants indicated that they are aware of a product line strategy in their organization (statement #1). We, thus, conclude that there are some general announcements in the organization regarding product lines, in the form of announced goals only, without the actual definition and enactment of mechanisms to implement these goals.

Since organizations regularly define work procedures and other governance mechanisms to steer the development process, we suggest to increase the awareness to the cloning activity as a reuse method and to merge it into current processes including shaping the role schema, refining the work procedures, adding basic measures to be able to assess its implementation, and educating about the risks and trade-offs of cloning. This way, evidence can be collected about when cloning is a successful practice and when it is not, compared to other alternatives.

Examining reuse costs, we conjecture that there is a non-negligible investment in setting up a structured product line. In domain engineering, there is a cost in making the software component generic and reusable. In application engineering, there is a cost in retrieving the reusable component and configuring it for the specific product [18]. Organizations might face difficulties to raise the funds to pay these costs. However, there is a trade-off between short-term savings obtained through cloning and the long-term maintenance problems that are caused in the result. These should be taken into account when deciding on a reuse strategy. In organizations with loose governance, people might be tempted to "locally optimize" their work, thus, sacrificing the "globally-optimal" result.

V. Threats to Validity

External validity: Our main threat to external validity is the limited number of subjects stemming from six product lines. We attempted to mitigate this threat by approaching development organizations from different industrial sectors and by interviewing practitioners holding a spectrum of organizational roles. Furthermore, we acknowledge this threat by carefully avoiding to generalize our results. Instead, we invite other researchers to confirm or refute our findings by studying further companies.

Our exclusive data source are interviews and questionnaire responses. In particular, we did not perform any artifact study of the actual product line projects, for example, to analyze sizes of clones or their extent of replication. Such an analysis could complement our study and aim at cross-checking results. However, artifact studies are, in general, very difficult to perform for commercial, closed-source projects.

We limited our subjects to those who apply code cloning for product line development. Thus, estimating the frequency of this technique in practice is out of our scope. Due to our experience, we speculate that code cloning is a common approach to product line realization; however, a widely-distributed follow-up questionnaire would be necessary to empirically confirm (or refute) this hypothesis.

Internal validity: We see two main threats to internal validity. First, we might have misphrased some interview questions in a way that affects participants' answers, especially since participants are usually not aware of product-line-specific terminology. We mitigated this threat by doing pretests and refining our interview guide when questions raised confusion. Second, we might have misinterpreted participants' answers and derived incorrect conclusions, threatening the reliability of our study. This reliability primarily depends on the categories and concept hierarchy we created. However, these concepts were carefully discussed and verified against the transcripts by another author.

VI. Related Work

Software cloning—duplication and reuse with or without modifications—has seen active research in the last decade [19], both in the context of single systems development and software product lines. We now discuss such related work.

Clone Management Techniques. Clone management techniques have been extensively studied; in particular, *clone detection*—identifying cloned code artifacts [20], *clone synchronization*—tracking clones during evolution and propagating changes [21], [22], *clone merging*—integrating multiple changes that happened simultaneously to the clones on both sides [23], *clone correction*—eliminating clones through refactoring [24], [25], and *clone prevention*—assisting developers while writing code [26].

Our work differs from those, as we do not suggest any technique, but rather provide the basis to apply such techniques, by investigating cloning practices in industry. Further, we empirically investigate cloning in the context of software product lines, not single systems development.

Studies on Software Cloning. Although clones were initially considered undesirable [27], multiple authors later argued that this assumption is not necessarily true and reported corresponding empirical data. Kapser and Godfrey [28] describe several observed patterns of cloning and provide evidence that cloning is used as a principled engineering technique. Aversano et al. [29] present empirical data on how clones are maintained. The results seem to indicate that the majority of clones is maintained consistently, and if an inconsistency is introduced, then often intentionally with a reason. This finding was confirmed by Göde and Koschke [30], who study the frequency of changes to clones, concluding that only 12% of clones were ever changed and only 15% of all changes were unintentionally inconsistent. Kim et al. [31] further indicate that eliminating clones not necessarily improves the development. By analyzing clone genealogies from codebase histories, they conclude that for short-lived clones, the removal effort is often too high to pay off, whereas long-lived clones are usually too hard to refactor. Ernst et al. [6] study forking of open source projects, which can be seen as a special form of inter-organizational cloning by putting whole codebases under another leadership. Forking allows addressing new requirements, but at the risk of fragmenting developer communities. Although our work focuses on intra-organizational cloning in the context of SPLE, such studies complement our finding that cloning is considered a beneficial and sometimes necessary technique.

Similar to our findings, Cordy [16] reports that developers enjoy the freedom of making arbitrary changes to their clones. The reported experience from projects in the financial domain shows that clones increase the freedom of developers while reducing coupling, testing costs, and maintenance risks. Eliminating clones (i.e. sharing code) increases coupling between modules, thus, changing such shared code requires re-testing of all modules using it. Since in these financial projects, most costs went into testing (70%), eliminating clones would have raised costs significantly—assuming frequent changes to clones. Unfortunately, Cordy's work—as he points out—is only a personal experience report, not an empirical study. This work also does not consider the PL perspective.

Cloning in the Context of SPLE. Several research works [32], [4], [33], [5] indicate the existence of industrial product lines that are realized by cloning instead of using variability management techniques. Faust and Verhoef [4] further describe "Software Mitosis"—the uncontrolled adoption of systems in a global organization, where successful systems are duplicated and modified by local sub-organizations.

Staples et al. [34] indicates that Software Configuration Management (SCM) systems, such as CVS, RCS or SVN, are used to realize product lines by exploiting their branching and merging capabilities. The authors report experiences on creating a product line using an SCM system and challenge some tenets from the literature, such as the need for a complete upfront scoping process and a variability-enabled architecture [35]. Van Gurp et al. [36] confirm this perspective and argue that SVN can be sufficient for SPLE. Even specific product line tools based on SCM systems have been developed, such as Thao et al.'s [37] MoSPL tool, which has built-in product derivation support.

Although our study was inspired by all these singular reports, we not only confirm the application of cloning in SPLE contexts, but go significantly beyond by qualitatively investigating the cloning practice. We also observed the use of SCM tools. In fact, four of our participants stated that they create a branch for each new product. However, studying whether SCM systems improve the cloning practice was out of our scope, but would constitute interesting future work.

Re-engineering of Products into Product Lines. Based on the assumption that many product variants are clones of others, many researchers provide solutions for re-engineering similar products into a configurable product line. The approaches of Faust and Verhoef [4], Mende et al. [33], [5], and Frenzel et al. [7] all identify common functionality using clone detection techniques with certain metrics in order to lift sufficient similarity to the architectural level. Jiang et al. [8] report industrial experience on maintaining and evolving a mobile phone product line at Motorola. Facing increasingly time- and resource-intensive, but quality-degrading evolution, they discuss data mining techniques to maintain reuse and to reduce design erosion. Ryssel et al. [9] detect cloned subsystems across Simulink models and re-integrate them into one subsystem with variation points. These variation points and their dependencies are identified using formal concept analysis and are further used to generate a feature model. Rubin and Chechik [10] introduce a framework based on model comparison and merging to refactor cloned product models into an annotative SPLE representation. Yoshimura et al. [11] present an alternative approach, which detects variability in software products from change history, assuming that each product consists of individual components and evolution entails an update to these. Duszynski et al. [38] describe a framework for the analysis and visualization of similarities across related systems. After identifying corresponding files, the framework facilitates browsing variants in large code bases.

While the techniques above can be used to eliminate clones towards systematic variability management, our work aims at improving the understanding on whether and when clones should be eliminated. Therefore, we investigate the underlying rationales, advantages, perils, as well as organizational and governance aspects of cloning.

Maturity of SPLE. Our results can further be compared to existing work on organizational aspects and maturity levels of SPLE, such as Riva and Del Rosso [32] or Bosch [39]. These authors classify the clone-and-own approach as the lowest level of maturity in an organization. Our study challenges this simplification, at least from the perception of our participants. As cloning appears to be legitimate in many cases, even organizations with very mature development processes might intentionally apply cloning. Whether existing maturity levels need to be refined requires further research, however.

VII. Conclusions

SPLE reuse approaches promote development of related software products from a common set of assets in a prescribed manner. Yet, developers still use cloning to realize different products of a product line. Without understanding the nature of the current cloning practices, attempts to transform collections of cloned products into representations promoted by contemporary SPLE approaches, as well as attempts to improve the approaches themselves, are difficult and unfocused.

We thus conducted a study to collect empirical data on the current development practices in organizations that employ cloning to realize product lines. We believe that acquiring better understanding on why and how cloning is practiced in industry can help developing better methodologies, tools, and measurement models that promote efficient software reuse.

Our study involved eleven participants from six cloned product lines of three well-established large-scale organizations. We observed human, organizational and technical aspects that are part of the cloning culture, discussed main characteristics of cloning and highlighted perceived advantages and disadvantages of the approach. We believe that our observations hold for many similar organizations.

We observed that the majority of interviewed practitioners involved in developing cloned product lines perceive cloning as a favorable reuse approach, which is "natural" to developers and has a low entrance barrier. Our participants stated that cloning is a *rapidly* available mechanism, which allows them to start from an already *specified and verified* functionality, while leaving the *freedom and independence* to make any necessary modifications to it. However, in the long run, cloning might result in difficulties to perform maintenance and evolution tasks.

Trade-off between savings obtained through cloning and the longer-term problems introduced by it should be further investigated. Yet, it became clear that any approach attempting to re-engineer cloned product lines into structured SPLE models should clearly show and quantify benefits in doing so, as practitioners' desire to abandon their current cloning practices is not obvious. Moreover, structured SPLE reuse approaches should strive to maintain those qualities of cloning that were perceived as important by the industrial practitioners. The SPLE approaches should also come up with clear measurement models, quantifying improvement achieved by SPLE adoption.

In addition, lack of organizational roles responsible for and promoting software reuse raises the need for establishing organizational structures that support SPLE. Industrial practitioners should also be educated on different reuse approaches, focusing on expected benefits from adopting those both on the personal and on the organizational level.

Acknowledgements

We thank all the developers, architects, technical learners and QA engineers who participated in our study and offered their valuable feedback.

References

[1] P. C. Clements and L. Northrop, *Software Product Lines: Practices and Patterns*, ser. SEI Series in Software Engineering. Addison-Wesley, 2001.

[2] H. Gomaa, *Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures*. Addison Wesley, 2004.

[3] K. Pohl, F. Guenter Boeckle, and van der Linden, *Software Product Line Engineering : Foundations, Principles, and Techniques*. Springer, 2005.

[4] D. Faust and C. Verhoef, "Software Product Line Migration and Deployment," *Software: Practice and Experience*, vol. 33, no. 10, pp. 933–955, 2003.

[5] T. Mende, R. Koschke, and F. Beckwermert, "An Evaluation of Code Similarity Identification for the Grow-and-Prune Model," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 21, no. 2, pp. 143–169, 2009.

[6] N. A. Ernst, S. M. Easterbrook, and J. Mylopoulos, "Code forking in open-source software: a requirements perspective," *CoRR*, vol. abs/1004.2889, 2010.

[7] P. Frenzel, R. Koschke, A. P. J. Breu, and K. Angstmann, "Extending the Reflexion Method for Consolidating Software Variants into Product Lines," in *Proc. of WCRE'07*, 2007, pp. 160–169.

[8] M. Jiang, J. Zhang, H. Zhao, and Y. Zhou, "Maintaining Software Product Lines - an Industrial Practice," in *ISCM'08*, 2008, pp. 444 –447.

[9] U. Ryssel, J. Ploennigs, and K. Kabitzsch, "Automatic variation-point identification in function-block-based models," in *Proc. of GPCE '10*, 2010, pp. 23–32.

[10] J. Rubin and M. Chechik, "Combining related products into product lines," in *Proc. of FASE'12*, 2012.

[11] K. Yoshimura, F. Narisawa, K. Hashimoto, and T. Kikuno, "FAVE: Factor Analysis Based Approach for Detecting Product Line Variability from Change History," in *Proc. of MSR'08*, 2008, pp. 11–18.

[12] B. Glaser and A. Strauss, *The Discovery of Grounded Theory: Strategies for Qualitative Research*, ser. Observations (Chicago, Ill.). Aldine de Gruyter, 1967.

[13] J. Corbin and A. Strauss, *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*, 3rd ed. Sage Publications, Inc., 2008.

[14] S. Adolph, W. Hall, and P. Kruchten, "Using Grounded Theory to Study the Experience of Software Development," *Empirical Software Engineering*, vol. 16, pp. 487–513, 2011.

[15] M. Q. Patton, *How To Use Qualitative Methods in Evaluation*. Newbury Park, CA: Sage., 1987.

[16] J. R. Cordy, "Comprehending Reality - Practical Barriers to Industrial Adoption of Software Maintenance Automation," in *Proc. IWPC'03*, 2003, pp. 196–.

[17] B. Nuseibeh, "Ariane 5: Who dunnit?" *IEEE Software*, vol. 14(3), pp. 15–16, 1997.

[18] G. Böckle, P. Clements, J. D. McGregor, D. Muthig, and K. Schmid, "Calculating ROI for Software Product Lines," *IEEE Software*, vol. 21(3), pp. 23–31, 2004.

[19] R. Koschke, "Frontiers of Software Clone Management," in *Frontiers of Software Maintenance, 2008.*, 2008, pp. 119–128.

[20] S. Bellon, R. Koschke, G. Antoniol, J. Krinke, and E. Merlo, "Comparison and Evaluation of Clone Detection Tools," *IEEE Trans. Software Eng.*, vol. 33, no. 9, pp. 577–591, 2007.

[21] M. de Wit, A. Zaidman, and A. van Deursen, "Managing Code Clones Using Dynamic Change Tracking and Resolution," in *Proc. of ICSM'09*, 2009, pp. 169 –178.

[22] E. Duala-Ekoko and M. P. Robillard, "Clone region descriptors: Representing and tracking duplication in source code," *ACM Trans. Softw. Eng. Methodol.*, vol. 20, pp. 3:1–3:31, July 2010.

[23] T. T. Nguyen, H. A. Nguyen, N. H. Pham, J. M. Al-Kofahi, and T. N. Nguyen, "Clone-Aware Configuration Management," in *Proc. of ASE'09*, 2009, pp. 123–134.

[24] M. Balazinska, E. Merlo, M. Dagenais, B. Lagüe, and K. Kontogiannis, "Advanced Clone-Analysis to Support Object-Oriented System Refactoring," in *Proc. of WCRE '00*, 2000, pp. 98–.

[25] M. Rieger, S. Ducasse, and G. Golomingi, "Tool Support for Refactoring Duplicated OO Code," in *Proc. of ECOOP'99 Workshop on Object-Oriented Technology*, 1999, pp. 177–178.

[26] B. Lague, D. Proulx, J. Mayrand, E. M. Merlo, and J. Hudepohl, "Assessing the Benefits of Incorporating Function Clone Detection in a Development Process," in *Proc. of ICSM'97*, 1997, pp. 314–.

[27] S. Jarzabek and S. Li, "Unifying Clones with a Generative Programming Technique: a Case Study," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 18, pp. 267–292, 2006.

[28] C. J. Kapser and M. W. Godfrey, ""Cloning Considered Harmful" Considered Harmful: Patterns of Cloning in Software," *Empirical Softw. Eng.*, vol. 13, pp. 645–692, 2008.

[29] L. Aversano, L. Cerulo, and M. Di Penta, "How Clones are Maintained: An Empirical Study," in *Proc. CSMR'07*, 2007, pp. 81–90.

[30] N. Göde and R. Koschke, "Frequency and Risks of Changes to Clones," in *Proc. of ICSE'11*, 2011, pp. 311–320.

[31] M. Kim, V. Sazawal, D. Notkin, and G. Murphy, "An empirical study of code clone genealogies," in *Proc. of FSE'05*, 2005, pp. 187–196.

[32] C. Riva and C. Del Rosso, "Experiences with Software Product Family Evolution," in *Proc. of IWPSE'03 workshop on Principles of Software Evolution*, 2003, pp. 161–.

[33] T. Mende, F. Beckwermert, R. Koschke, and G. Meier, "Supporting the Grow-and-Prune Model in Software Product Lines Evolution Using Clone Detection," in *Prof. of CSMR'08*, 2008, pp. 163 –172.

[34] M. Staples and D. Hill, "Experiences Adopting Software Product Line Development without a Product Line Architecture," in *Proc. of APSEC'04*, 2004, pp. 176–183.

[35] J. Bosch, "Maturing Architectures and Components in Software Product Lines," in *Component-Based Software Quality*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2003, vol. 2693, pp. 246–258.

[36] J. Van Gurp and C. Prehofer, "Version Management Tools as a Basis for Integrating Product Derivation and Software Product Families," in *Proc. of the SPLE'06 Workshop on Variability Management*, 2006, pp. 48–58.

[37] C. Thao, E. Munson, and T. Nguyen, "Software Configuration Management for Product Derivation in Software Product Families," in *Proc. ECBS'08*, 2008, pp. 265–274.

[38] S. Duszynski, J. Knodel, and M. Becker, "Analyzing the Source Code of Multiple Software Variants for Reuse Potential," in *Proc. of WCRE'11*, 2011, pp. 303–307.

[39] J. Bosch, "Maturity and Evolution in Software Product Lines: Approaches, Artefacts and Organization," in *Software Product Lines*. Springer, 2002, vol. 2379, pp. 247–262.