UNIVERSITY OF GOTHENBURG

Shared tools in software organizations

An empirical investigation

Bachelor of Science Thesis in Software Engineering and Management

DIMITRIOS PLATIS
JIAXIN LI

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
Göteborg, Sweden, June 2016

Shared tools in software organizations
An empirical investigation

Dimitrios Platis
Jiaxin Li

Examiner: Michel Chaudron

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Cover image source: http://www.xda-developers.com/custom-toolchain-roms-kernels/

Department of Computer Science and Engineering
Göteborg, Sweden June 2016

Shared tools in software organizations: An empirical investigation

Dimitrios Platis
University of Gothenburg
Software Engineering & Management BSc
dimitris@platis.solutions

Jiaxin Li
University of Gothenburg
Software Engineering & Management BSc
lijiaxin1012@gmail.com

Abstract

In the present research, tools and toolchains were investigated through the perspective of being shared within software organizations. Particularly, we conducted a literature review which was combined with a case study, that took place in two software organizations, in Gothenburg, Sweden. The different implications of shared tools and toolchains were studied and especially, their benefits, challenges and suggested characteristics. The important role of interoperability was verified, as did the suitability of a generic software assessment method, to be applied on shared tools. The paper suggests that further work can be conducted on the subject, in order to refine the assessment method and investigate shared tools in ecosystems with external actors.

1 Introduction

During the last years, the effect of software and the related products in our daily lives is ever increasing, with the software development industry currently playing a significant role in the economies of many countries around the globe. As the value and complexity of the software products increase, so does the importance of the programming tools to build them. Tools and, their combination, toolchains are employed in order to facilitate software development and enable the integration of different components often developed by separate teams.

These tools and toolchains, often need to be shared across a software organization, in order to facilitate the development process by having a common workflow and seamless communication. However, previous studies suggest [6] that despite knowledge sharing is considered important in the industry, most of the secondary software, such as tools and toolchains, are developed in-house and seldom shared with the rest of the organization or the extended ecosystem. Additionally, there appears to be a scarcity of assessment methods on tools [18, 8], in order to determine their appropriateness for various contexts.

It has been suggested that companies benefit from tools being shared systematically and that more investigation needs to be done on the matter [37]. Based on the identified gaps, we designated the implications of shared tools and toolchains as the topic of our investigation. This will be achieved by adopting a combination of a literature review with a case study. Particularly, this paper elaborates on their benefits and challenges in software organizations, attempts to establish several guidelines as to which attributes shared tools should be characterized by and ultimately propose an assessment method on them, based on an existing framework. Towards achieving this goal, we have devised a quartet of research questions. Specifically, the main research question revolves around a method on how to assess shared tools. This could sway the decision regarding which shared tool or toolchain should be adopted or to identify how to improve already existing ones. Moreover, the secondary questions consist of inquiries about the advantages and obstacles commonly accompanying shared tools and toolchains, as well as the intended characteristics of tools when shared in a software organization.

In section 2 previous studies can be found on the theoretical background of this work, as well as on relevant topics. Next, in section 3, the research questions are outlined, followed by section 4 where the adopted methodology to investigate the various topics is illustrated. Afterward, a reader can find the analysis of the collected data, while the resulting discussion is located in section 6. Finally, this paper elaborates on the various validity threats that were encountered, as well as the deployed measures to mitigate them, with the epilogue in section 8, summarizing the various findings and suggesting topics for future research.

2 Background and related work

In order to facilitate the interpretation of the collected data, it is necessary to establish the theoretical foundations of this study. This is achieved, by outlining the required background knowledge on the subject, such as various defi-

nitions on tools and the different kinds of tools in existence, as well as related work on the subject of shared tools in software organizations.

2.1 Background

Tools are software artifacts that support a large number different software engineering processes. They could be involved in various stages in a product's life cycle, from development and testing to integration and deployment.

They, are typically programs or scripts of lower complexity compared to the primary software they are supporting and can be used in conjunction with other tools in order to fulfil a purpose or to carry out a specific function. Tools can be either stand alone and explicitly launched by the user, or belong to a broader platform such as an Integrated Development Environment (IDE) in order for their execution and combination to be more seamless.

Toolchains, on the other hand, refer to a number of separate and stand alone tools, that are "chained" together, so the output of one becomes the input of the next, in order to perform complicated tasks or produce a collection of more complete software artifacts. The individual tools that are comprising a toolchain can, but are not required to, be utilized in sequence in order for their combined utilization to accomplish a more advanced goal. A characteristic example of a toolchain supporting development could involve the compilation, assembling and linking of source code, therefore using a set of three distinct tools, for a particular runtime environment.

In the context of embedded systems, Biehl (2013) mentions that their development frequently involves a variety of tools. As a consequence, in order to accomplish a holistic tool support throughout the development process, the toolchains were introduced, to combine the various tools [12]. The author argues that toolchains have evolved from unsystematic in-house single-purpose software, to complicated and advanced systems, in order to facilitate a distributed development process, integration and communication between diverse artifacts.

Tools can support software development, such as source code editors, assemblers, linkers and compilers. Modern tool suites and programming languages are also accompanied by a debugger. In Model Driven Engineering (MDE), tools have a very high importance, due to the fact that model transformation plays a major role in enabling the development at a higher abstraction level (i.e. models) to be realized as a platform specific implementation [28].

Particularly, in MDE it is required for the tools to be able to aid in automating the various model transformations that are commonly a necessity. According to Sendall and Kozaczynski (2003), such tools should not be limited to providing a predetermined set of model transformation, but allow technically skilled users with a degree of customizability to the transformations the tool can perform. On top of this, they consider as a positive trait of an MDE tool one that can recognize the particular context and offer various proposals on what model transformation should be applied. As model transformation implies the introduction of a model as an input to a tool and the production of one or more models as the tool's output, MDE toolchains are often created. In fact, researchers suggest [29] that the maturity of such a toolchain is needed in order to leverage the advantages of MDE.

Furthermore, in the category of tools that support software development, we could include version control tools (e.g. Git), tools text search tools such as grep and tools that generate code from visual artifacts, as in the case of graphical user interfaces (i.e. WindowBuilder in Eclipse) and instruments to monitor performance or possible bottlenecks, such as profilers.

Tools can also be used for testing with a plethora [27] of the specific kind in circulation. Pohjolainen (2002) has created a categorization of testing tools into the field that they are applied, noting that those categories are often overlapped. The primary classification includes tools used in order to design the tests, test the GUI or tools that facilitate the management of tests. Specifically, these categories include test case and data generators, automated tests for GUI centric artifacts and automation tools for applications without graphical interfaces.

Additionally, there are tools that provide static code analysis, with lint being a well-known example and tools that enable the user to implement tests and test evaluation tools, that facilitate the quality assessment of the tests, such as code coverage. Furthermore, the author has also outlined tools that enable their users to perform integration, regression and requirements testing. Mustafa et. al. (2009) also offer another taxonomy of testing tools [30], based on different test tool attributes, note that the majority of the testing tools they discovered through various sources, were intended for web applications.

Furthermore, as agile practices gain a foothold, integration and deployment tools acquire more significance. Particularly, it is a common challenge [31] in an industrial context to developing software at different locations, that is being deployed and tested for various hardware. In order to make the continuous integration methodology more lean and productive, the build and testing processes need to be automated. This is where continuous integration tools, come into play. Jenkins is a characteristic example of the said tools and its main feature involves the execution of a predetermined set of instructions (i.e. "jobs") triggered by a specific event, such as a commit using a version control tool or time intervals. Complementary to the above, Jenkins offers test reports and notifications on the build process to the involved developers. One particular advantage of the

tool, beneficial to the industry is its ability to commence jobs remotely, on several locations, with different hardware specifications and operating systems.

Another relevant technique, called continuous delivery promotes the production of software in compact cycles. This approach qualifies the software as constantly releaseable, therefore offering the organizations that follow it, the advantage of deploying updates and upgrades to their live systems or services rapidly, therefore subsequently gaining an upper hand against their competitors [32]. A tool that is inspired by the continuous delivery principles, is Go by ThoughtWorks. Go common tasks, are added as "pipelines" while the instantiations of the pipelines are named "jobs". Go additionally offers a visualization of the process, which enables the users tp locate the current progress of the process, from committing new code to the deployment of its changes.

2.2 Related work

Previous research on the topic of assessing shared tools and toolchains has not been conducted to our knowledge. However, there have notably been works on assessment of different or similar software artifacts shared in the context of software ecosystems. Additionally, we have identified previous studies on the characteristics and assessment of tools and toolchains.

2.2.1 Software Ecosystems

As software ecosystems are emerging and increasingly gain significance within the Software Engineering field [1, 2] it is feasible to compare them along with their common implications and characteristics, to that of large-scale and decentralized software products inside a single organization.

This correlation, is backed by researchers such as Schultis et. al. (2014) who in their research are referring to them as "internal software ecosystems" [33] and have conducted an extensive case study in order to discover the relevant architectural challenges that exist in such settings. Particularly, they claim that the lack of a centralized structure, in large software projects developed within a single organization, creates several architectural concerns. They conducted a case study, which permitted them to generate among others a categorization of typical architectural challenges, along with a detailed insight and metrics, of the various collaboration models they have discovered. As such collaboration models are characteristic, they can be used in order to identify the current situation and help researchers or managers decide on corrective measures, in order to improve the general development process.

More on the architectural challenges of software ecosystems, Bosch (2010) mentions the various challenges and their proposed solutions, that are surfacing in regards to architecture which are specifically caused by the involvement of additional "external" developers, who add new features [7]. Notable concerns include low interface stability, cumbersome overview of the evolution, security and reliability issues. Resemblance of Bosch's research to the current research on tools shared within software organizations can be spotted, due to the fact that many developers, often on different sites have to work on the same extended product or platform, which can contain legacy code and the need for decisions to be taken, that will have a very broad impact and have to determine a general direction, often arise.

Next, Hammouda et. al. (2015) discuss the implications of maintaining APIs in a software ecosystem that is constantly changing and evolving [4] therefore, need to satisfy the ecosystem's vision. The authors stress the challenges of API design within an ecosystem and argue about the necessity to steadily oversee the status, in order to identify plausible reasons for reassessment. They propose the "Ecosystemability Assessment Method" as the technique to evaluate APIs under the scope of a software ecosystem. As large scale products within a single software organization can be considered as internal software ecosystems, the particular assessment method of APIs can offer us valuable insights and inspiration on how to design an assessment method for shared tools.

Cao et. al. (2015), on the topic of assessing APIs from the perspective of software ecosystems, have introduced a strategy to evaluate APIs from the viewpoint of the users, by conducting a case study. Specifically, based on a combination of "fitness dimensions" as well as technical and cognitive profiles of API users, they proposed a new method of API assessment. Their work and in particular their methodology, namely utilizing an already existing framework which based on case study is customized for a specific purpose, can be leveraged for the current research on assessing shared tools.

Furthermore, the study by Sekitoleko et. al. (2015) provides valuable insight and findings on secondary software development in Model Driven Engineering, as it was investigated through a case study conducted within automotive ecosystems [6]. Based on the definition of the author, on what is secondary software, it is inferred that tools and toolchains that are the subject of this present study, fall within their scope. Therefore, the results of that particular study maintain a high degree of relevance and semantic correlation with this one. To be more specific, Sekitoleko et. al. support that despite the undeniable benefits of Model Driven Engineering's to the industry, the absence of expertise on secondary software, these benefits cannot be easily leveraged. Their study explored the various sociotechnical aspects of secondary software development and ways to improve the development of such software artifacts, in order

to decrease the necessary time for software development in the ecosystem of automotive industries. Eventually, their study stresses the need for the establishment of a systematic structure that enables both development and sharing of secondary software, which would also decrease common challenges expected in Model Driven Engineering, such as low interoperability, between the various software entities. The results of this study, are highly relevant to the current paper, as it discusses the development of very similar software artifacts within automotive software ecosystems. It provides us with the foundations to begin elaborating on shared tools and their various implications.

2.2.2 Tool assessment

In regards to considerations about tools and their characteristics, Barth et. al. (2012) elaborate on the significance of interoperability and seamless communication between tools and regardless of organizational restrictions on location and development phase [8]. The authors note that despite the importance of openness and interoperability being widely recognized, it is not realized at a high degree by the related software that is available in the market. They proceed to suggest an assessment technique for tools, judging on their interfaceability with other tools. Additionally, they suggest the relevant criteria and metrics, for this assessment to become systematic, in order for the users to be able to determine the appropriateness of the various tools at hand, while the developers can create tools based on those guidelines. The importance of interoperability and particularly the significant role it is suggested to play when combining tools, will help us define our assessment criteria, on shared tools within software organizations.

Schlegel et. al. (2010) present a prime paradigm of a model driven engineering toolchain, devised for an environment that inherently hinders interoperability [9]. In particular, they suggest a model driven engineering approach which focuses on the robustness of robotic systems. They have accomplished that by creating a robotics metamodel, accompanied by a preliminary set of non functional characteristics and requirements. Next, they established a toolchain, which enabled them to reap the benefits of model driven engineering. Specifically, it is utilized for its model transformation and code generation capabilities, as well as the means to progress towards resource awareness, e.g. through providing analyses of the scheduled real-time activities. The authors' example, despite being domain specific, can be used in order to provide us ground for generalization and the basis for constructing guidelines and best practices on toolchains.

Complementing the above literature, as to the reliability of tools, Wildmoser et. al. (2012) propose a systematic technique to evaluate tools and determine "tool confidence levels" according to the ISO 26262 standard [15]. In the automotive industry, as well as other domains, there are very strict requirements on the safety and failure rate of the developed software artifacts, including tools. The authors, based on empirical observations from large projects in an industrial context, have devised a strategy that systematically determines tool confidence levels, which enables vendors to disqualify tools that are not up to par with the ISO quality standard. The authors elaborate on the means to identify failures and propose two techniques in order to accomplish that task. A function based as well as an artifact based. Additionally, they devised a tool that provides the necessary analysis, to be used in conjunction with those strategies and to determine their ability to provide results of high significance. Robustness and reliability are admittedly important aspects when assessing a tool. However, the assessment of tools shared in software organizations will need to take more aspects into consideration.

Furthermore, toolchains are argued to positively affect productivity and facilitate cost reduction. In this regard, Biehl et. al. in [10] and [11], discuss the benefits of toolchains on productivity despite their relatively high implementation cost. They attempt to quantify the decision making on the adoption of a toolchain by applying a cost analysis model in order to estimate the costs involved in the realization of a toolchain, therefore calculating their efficiency in regards to the capitals necessary to invest. Their theory was verified via a case study in an industrial context. Cost, is a significant factor of tool assessment, especially since it can be interrelated to the usability of a tool or a toolchain.

A number of usability and functionality attributes of tools used in the automotive electrical and electronic architectures were quantified and evaluated on the grounds of interoperability and compatibility in order to form a toolchain, in the work by Waszecki et. al. (2013). Particularly, they recognize the difficulty of selecting the most suitable tools to support a specific development process within an automotive corporation [18]. This process should not merely involve the implementation phase, but modeling and testing as well. Moreover, they argue that even if that is accomplished, combining them into an efficient toolchain that covers the various stages of the development process, is not guaranteed and remains a risk. They propose a systematic approach towards composing such toolchains, by providing an overview of the relevant industrial design processes, its related challenges and investigating a number of commercially available tools.

2.2.3 Tool characteristics

To begin with, Whittle et. al. (2013) [36] designate tools as one of the deciding factors involved in the adoption

of model driven engineering in an organization. The authors consider as their key contribution a classification of aspects which represent the impact of tools on the adoption of model driven engineering. Their taxonomy includes four primary categories. The technical factors, the internal and external organizational ones and finally the social. There, different challenges are mentioned, such as their high complexity along with low usability, incompatibility with an organization's structure and culture, considerable cost, scarcity of knowledge on how to select them and finally, the reluctance of using the tools, due to lack of trust in their abilities. Our paper builds upon this research, in order to define a set of common challenges for tools that will suggest the best practices regarding their characteristics and ultimately an assessment method.

On building toolchains, Wolvers et. al. (2013) stress the problems caused by the inherently different meta-models among the different tools [38]. They propose an integration framework, that is found on the Open Services for Lifecycle Collaboration approach. To verify their suggestions, they conducted a case study in an industrial context and developed an adaptor to interface the various tools as web services.

Porter et. al. (2009) intrigued by the need to create overall better toolchains that would, in turn, increase the productivity of the developers of embedded systems, introduced a model based prototyping toolchain, accompanied by a hardware in the loop system, which would enable embedded designers to evaluate concepts with higher efficiency [13]. Particularly, common problems they mention in the contemporary tools (e.g. Simulink) include the lack of model representation of several crucial parts of the system, absence of modeling features for the deployment activities and loosely integrated verification tools. Their toolchain aims to counter those disadvantages by offering modeling options for both functional and nonfunctional properties of the system, as well as tighter integration of verification and analysis tools.

Next, Biehl (2013) considers the task of creating toolchains that seamlessly combine the various tools for the development of embedded systems, laborious and time-consuming [12]. The researcher attributes this to the fact that it is, to a considerable extent, a non-automated task. To hinder the process further, toolchains are largely described using generic modeling languages or languages non-specific to the embedded systems domain. This, in turn, increases the challenges involved in their development and in order to tackle this, the author has proposed a domain-specific modeling language, for the elaboration of toolchains. Moreover, Burden et. al. (2014) also point out the large amount of effort, required to integrate tools in the process of a software organization, as they need to be customized to a large extent first or the process to be altered in order to support them [26]. The difficulty to implement a toolchain combined with their importance in the development of embedded systems, designates it as one of the most important challenges also expected in the context of shared tools.

Furthermore, Karsai et. al. (2006) are introducing a set of model-driven engineering tools in order to solve the low verifiability and maintainability, of IVHM systems which were developed using hand written code [19]. The researchers, focus their efforts in the reusability of their tool suite, but more importantly imply the importance of interoperability in order to facilitate integration, when developing a component of a larger system. Reusability is anticipated to gather an even higher importance under the scope of shared tools.

Voget et. al. (2010) in their work, introduce various concepts in order for a new and improved toolchain in the automotive industry to emerge, based on the AUTOSAR standard [34]. Specifically, they recognize the importance of sharing and exchanging models, within an automotive software organization, a factor which for the authors directly implies that interoperability of tools is a factor the success of the AUTOSAR methodology depends on. Additionally, they stress that common challenges derive from the insufficient import and export features of various tools. Moreover, the origins behind many impediments in the process of creating a seamless toolchain and workflow can be identified in the fact that tools are frequently characterized by different, fundamentally incompatible, technologies, with steep learning curves.

Interoperability, or rather its lack, is also recognized by Kern and Kühne (2009) as one of the most prominent challenges in the existing commercial tools, as well as the main hindering factor for developing toolchains and reusing models [20]. In their work, they demonstrate their approach by increasing the interoperability between two modeling tools, Microsoft Vision and Eclipse Modeling Framework.

Finally, Walderhaug (2013) discusses ways to increase the usability of model driven engineering tools, from the perspective of the healthcare software industry [14]. Specifically, the author elaborates on a model development toolchain that was created, which aided in developing web applications and services based on the CEN-13940 standard. The toolchain was particularly designed in a manner that promotes the design of software artifacts with high interoperability. The author argues that the particular domain knowledge was satisfyingly embedded in the model driven engineering toolchain and the various resulting observations can be applied to model driven engineering toolchains, in general, regardless of the domain they are realized in. For example, those recommendations stress the need for the modelling tool to provide the project structure, as well as design model verification and validation. Qualities that are also highly evaluated by tools in the domain of embedded

systems, i.e. in [13]. More importantly, a collection of suggestions that was derived from the development of the health care toolchain, on how to apply domain-specific requirements into model driven development tools, is offered.

3 Research questions

The research questions will provide the guidelines and the central points of our research. They aim to indicate the various implications of shared tools and suggest an assessment technique.

To begin with, light will be shed upon the various advantages reaped by the software organization, from the adoption of shared tools. These will be formulated by synthesizing data from both the previous literature and data derived from the various interviews, conducted within the case study.

- RQ1 - What are the benefits of sharing tools in a software organization?

The next research question is oriented towards illustrating the challenges involved in a software organization that utilizes shared tools. The various issues that are commonly surfacing, will be derived from common problems discovered in the bibliography, as well as those designated by our interviewees.

- RQ2 - What are the challenges of sharing tools in a software organization?

Acquiring a response to the previous, enables us to get a comprehensive overview of the technical domain and the various inherent difficulties from sharing tools in software organizations, as well as their derived profits. Next, in order to define an assessment method for tools and toolchains shared in software organizations, the researchers have to formulate a picture on the distinct characteristics, tools should have, in order to be shared efficiently.

- RQ3 - Which should the optimum characteristics of shared tools be?

Finally, an attempt is made to define an assessment method for shared tools. This finding is comprised of the synthesis between the previous research questions and a generic software assessment method that is described in section 4.3. Particularly, the said assessment method will be verified in terms of applicability in the domain of shared tools. An assessment method should specify the appropriateness of a shared tool, for it to be adopted by the organization or improved if already being used.

- RQ4 - How should tools shared in a software organization be assessed?

4 Methodology

In the process of defining the benefits and challenges of sharing tools across a software organization, we conducted a bibliographic review on the subject and combined its findings with data gathered from an exploratory case study, which took place in Gothenburg, Sweden, involving employees from two software organizations. Particularly, in total eight individuals from two large software organizations, engaged in the automotive and communications domains gave the researchers a deeper insight on tool sharing, through six semi-structured interviews. The interviewees are either users or developers of different tools. Additionally, the BAPO framework [16] will be adopted in order to facilitate the analysis of the extracted data but more importantly was utilized in order to structure the interview guide.

Regarding the BAPO framework, an additional ”dimension” was added to the proposed ones, in order to increase the coverage and the applicability of the evaluation framework. Specifically, considering the nature of tools, which is to support another process, e.g. development, testing etc, we decided to enhance the evaluation framework with a ”product” perspective , in order to detect and illustrate the effects of shared tooling on the product that is being built, with the aid of the tools and toolchains.

4.1 Case study

In the process of extracting useful empirical findings from an industrial perspective, it was decided to conduct a case study in two large software organizations in the city of Gothenburg, Sweden. According to Höst et. al. (2007) there has been a plethora of paradigms of case studies conducted on software engineering topics, however, no specific to the field of software engineering, guidelines as to how they shall be realized. Therefore, they argue, the researchers intending to launch such initiatives in the software engineering field, can abide by the generic norms of case study rulebooks [22].

On the overall characteristics of case studies, Yin (1994) supports the idea that case studies have the ability to elaborate on technically specific circumstances with more unknown factors than data points [21]. Their results are commonly formulated based on multiple sources and after data triangulation. Additionally, data collection and analysis are modeled after the previous construction of the theoretical foundations, by the researchers.

As the core topic of this research has not been thoroughly investigated in the past and a plethora of aspects have remained relatively untouched by previous researchers, the case study that was conducted can be characterized as exploratory [17]. To put it differently, the study intends to discover more about the situation that has been developed

and search for new perceptions, ideas and hypotheses for further research.

The survey method [35] chosen to collect the data in this particular case study were semi-structured interviews. Half of the interviews were conducted on the working site of the employees while the rest in quiet rooms at the Lindholmen university Campus. As it is accustomed in semi-structured interviews which are common in case studies [17], the questions were predefined but not necessarily asked in a concrete order, depending on the flow of the conversation. If for example, an interviewee would elaborate on aspects that were covered on a question near the end, the course of the interview would be shifted, in order to discuss those concerns, without interrupting the subject's course of thought. In addition to this, complementary questions were in some occasions added or improvised, in order to investigate aspects that were not anticipated by the interviewers and no relevant question had been priorly scheduled.

Furthermore, in order to perform data triangulation so to validate and signify our findings, we chose two different software organizations for our data collection and extraction. Specifically, company A is engaged in the automotive industry, while company B is in the telecommunications domain. We opted for these two organizations based on the fact that they develop or use tools, that often need to be exchanged between different developer teams around the work site or even among different locations in the world. Additionally, they are interested in improving their process and sought after critical observations from external entities.

Motivated by the need to both investigate the topic and improve the research process simultaneously, we organized a pilot interview with three employees of company A. We utilized these preliminary findings in order to refine our interview guide. Particularly, we removed specific questions which were hard to comprehend and easy to misunderstand by our subjects, while introducing new questions, in order to clarify various concepts, to establish the profile of each interviewee and acclimatize them faster to the purpose of the discussion.

Next, a focus group discussion followed, with members of company A and the academia with many participants, on a broader context about common challenges encountered during the practical application of continuous integration in the automotive domain. This discussion enabled the researchers to get a better understanding of the general situation and increase their domain knowledge, which resulted in small modifications and improvements of the interview guide. The final interview with an employee from company A was arranged, in order to increase the data points from the automotive industry on the matter.

After a preliminary analysis of the collected resources from company A, we started to interview employees from company B, applying the knowledge and experiences that were attained from the initial interviews. Particularly, the accumulated data enabled the researchers to facilitate the comprehension of several domain specific terms and to make various parallelism to the context and different circumstances of company B.

In total, four employees from company B were interviewed, in separate interviews. The interviewees are employed in two different departments of company B, are all assigned to different roles professionally and use or develop different tools. This variety of roles and positions, enables the researchers to cover a larger spectrum of the software development activities taking place in the software organization and increase the validity of the results by minimizing biased opinions, as it might have been the case with subjects from a single department, working on the same domain and being tasked with adjacent roles. Interviews with more employees from both companies would have undeniably been beneficial, however, this was not achievable due to the limited available resources and time constraints.

Moreover, questions 4 and 5 from the interview guide, were used to answer RQ1, regarding the benefits of shared tools. The results from questions 6,7,8,9 were utilized for RQ2 regarding the challenges. The response to RQ3, as to the characteristics, was facilitated by interview questions 10, 11, 12, 13. Finally, in order to reply to RQ4 concerning the assessment method, no interview questions were exclusively used. In fact, it was based on data from the previous research questions, as well as the rest of the interview guide.

Last but not least, following the recommendations of Runeson et. al. (2009) the interviews contained three stages [17]. Initially, the interviewees and the interviewers were engaged in a light and introductory talk, about the goals of the research and its context in order to get acclimatized to both the researchers and the interview process. Next, the initial introductory questions were asked, when the subjects would talk generally about themselves and the role in the organization, as well as their definitions of various key terms. Finally, the attention is shifted and focused on the main topic of the discussion, on the sharing of tools and toolchains in their software organization.

4.2 Analyzing the data

Data that were collected from the surveys were transcribed and thematically organized in order to facilitate the analysis and extraction of findings. In order to progress towards the primary goal of this work, which is to generate an assessment method for tools shared within software organizations, the need to partly quantify and structure the collected data arises. To achieve this, we leveraged the BAPO framework and its various dimensions, by formulating the interview guide based on it, in order to facilitate the classification of data.

Next, the various categories according to which the data will be organized, have been carefully selected, based on the most common pattern the interviews would follow and are ultimately inspired by the questions that were asked overall, despite the various changes that were applied in the interview guide. In other words, despite the discussion not following an identical schema throughout the various different interviews, due to the various improvements, the essence of the guide remained the same. Therefore, it is applicable to attempt and group the findings from the various discussions with the employees of the software organizations.

This adopted categorization process was carefully selected and conducted, in order to enhance the traceability of results from the first-degree data sources, i.e. the interviews. A clear chain of the evidence was maintained for every conclusive remark that was made, so to increase the validity and significance of the study.

Data were classified for further analysis, according to the following semantic categories, followed up brief disambiguations:

1. Background: Data about the background of each interviewee and their definitions of key terms
2. Benefits: Data structured by the BAPO framework regarding the benefits of sharing tools
3. Challenges: Data structured by the BAPO framework regarding the challenges of sharing tools
4. Challenging factors and mitigation: Data on interviewees' responses to common challenges
5. Shared tools suitability: Data on the suitability of shared tools depending on each development phase
6. Characteristics for sharing: Data on the characteristics shared tools should have
7. Other attributes: Data on interviewees' opinions on common characteristics of shared tools

Each category found above was extracted from one or more survey questions, thus, the traceability can be ensured. The basic information section is composed of the basic information of the interviewee, the organization, the role of the interviewee in that organization and the data on the shared tool or toolchain in the organization. The benefits section will include the various benefits of sharing tools based on the BAPO framework along with an evaluation of the significance of each BAPO dimension by the interviewees. Likewise for the BAPO dimensions regarding the challenges.

Next, the challenging factors shall illustrate each subject's attitude towards some common factors mentioned in the literature. Data falling under this category, in regards to the challenges will be mapped to a 5-degree Likert scale of severity according to the interviewees' attitude towards them. In most occasions, the interviewees demonstrated a very clear verbal preference or disagreement with these aspects, therefore, we consider the quantification to a 5-degree scale plausible. Specifically, the scale begins with an attribute being most significant, represented by the number "1" and ends with the least significant features, being designated by number "5". The base or neutral degree is illustrated by number "3".

Furthermore, the category that follows will concern data regarding the key characteristics a sharing tool/tool-chain should have according to the interviewees, as well as the frequency that each characteristic was mentioned among all the interviewees. Eventually, the last category will hold data using a 5-degree scale, on the subjects' input on common attributes of shared tools, derived from the literature review.

This classification of the collected data, facilitated the extraction of findings and enabled us to have a firm overview of the collected data, by categorizing them in spreadsheets.

4.3 Evaluation Framework

The collected data from the interviews of the case studies were analyzed and correlated against each other, in order to discover common patterns, antitheses and eventually be used in order to create an evaluation method for tools. In regards to this, we have receded into using the criteria based assessment [23] by Jackson et. al. (2011). The technique has been devised in order to conduct software evaluations and therefore it is feasible, in concept, to apply it to a tool evaluation setting. In this work, it will serve as the basis of the shared tools evaluation method and will be modified accordingly, after utilizing the data from the case studies and the literature. This will enable us to specialize the method and narrow its application down from the generic software domain to the one of the shared tools and toolchains.

The criteria based assessment evaluates software on the grounds of usability, maintainability and sustainability. This method involves the auditing of the tools against various attributes and best practices that are typical of software, fulfilling those three quality requirements. The various assessment criteria that were adopted for the purposes of this study as well as their subcategories are presented below. Each of the criteria is accompanied by relevant questions, which enable the auditors to determine the level of compliance of the software in question, with the quality attribute.

1. Usability
    (a) Understandability
    (b) Documentation

(c) Buildability

(d) Installability

(e) Learnability

2. Sustainability & Maintainability

(a) Copyright/Licensing

(b) Community

(c) Accessibility

(d) Testability

(e) Portability

(f) Changeability

(g) Interoperability

As the authors of the criteria assessment methods stress in their work, the importance of each criterion can be adjusted to the specific case that the technique is being applied. To put it differently, it is often necessary to consider that each of the criteria should be considered with a different weight, or importance, in order to compensate the various special circumstances that arise from each special case. Subsequently, we utilized the results of the case studies, to assign weights to each of the quality attributes, in order to form an evaluation method for tools shared in a software organization.

Similarly, Jackson et. al. have designed another evaluation method, based on tutorials and focusing more on the various user personas perspectives [24]. Specifically, the tutorial based assessment offers an evaluation of a software artifact's usability as a log of experiences and observations, offering an insight from different user viewpoints. Moreover, we utilized its user-developer and developer perspectives, based on the fact that our interviewees commonly share most characteristics with those two profiles. The user-developer personas, provide software evaluations from the point of view of the ones that are developing against an API of a tool and use it in conjunction with other components. The said profile typically appreciates the easiness to install, configure and use a tool as well high interfaceabily with other components. On the other hand, the developer persona, commonly involves individuals who are either creating the software or modifying it and their usual concerns include the degree of difficulty to maintain, refactor or increase the functionality of a tool. Based on the above, we have devised the following list of questions, which will be used in conjunction with the above criteria, in order to assure that the coverage of the various concerns of the domain, is maintained at a maximum degree. Additionally, we propose the responses to be on a 5-degree scale, so to facilitate the assessment and comparison of different tools.

1. How easy is it to set up the environment, write and compile code for the tool or code that uses the tool? (1-Very easy, 5-Very hard)

2. How easy is it to specify which other tools or software and their versions are necessary in order to set up a development environment involving the tools? (1-Very easy, 5-Very hard)

3. How satisfied are you with the available tutorials and examples for the various tool versions? (1-Very satisfied, 5-Very dissatisfied)

4. How easy is it to understand the API documentation? (1-Very easy, 5-Very hard)

5. To what extent does the documentation cover the tool's API? (1-Completely, 5-Nominally)

6. How easy is it to specify the various intellectual property and copyright issues that may arise from the use of a tool? (1-Very easy, 5-Very hard)

7. How easy is it to understand the source code and/or design of the tool if it is available? (1-Very easy, 5-Very hard)

8. How easy is it to validate user code that involves the tool? (1-Very easy, 5-Very hard)

9. How easy it is to release or deploy user code that involves the tool? (1-Very easy, 5-Very hard)

The questions above should be used complementary to the ones proposed by each criterion of the tutorial based assessment, especially if criteria do not explicitly cover a certain factor, e.g. versioning and architectural design concerns.

4.4 Threats to validity

The validity of a research implies the extent to which its result can be trusted, are true, the various deductions that were made to reach them are valid and remain unaffected by the researchers' subjective perspective [17]. Threats to validity should be mitigated throughout the various stages of the research, from the literature review to data collection and analysis. The said risks, will be extensively discussed in Section 7. By adopting the classification scheme by Runeson et. al. (2009) [17] we can view the validity from four viewpoints and therefore categorize the various threats against it.

Particularly, the authors have defined the construct validity, which engages the issue of the alignment between the various "operational measures" of the study and what is actually the goal of the investigation (e.g. based on the

research questions). A threat to construct validity could be the lack of common perception between the interviewers and the interviewees, regarding fundamental terms, e.g. tools. Next, there is the internal validity which comes under debate when there are unprecedented factors that affect the phenomena and the researchers have not taken into account, while conducting their study. Internal validity could be compromised in case the persons conducting the research are causing biased answers, or unknowingly otherwise affecting the process and results.

Furthermore, the external validity revolves around the question whether the study's findings can be generalized and have a realistic reflection on different settings, outside the specific environment that was investigated. To put it differently, the external validity queries whether the results of the study are arguably universal and can be utilized by other researchers in similar settings. The main threats of this study are against the external validity, due to the fact that merely a limited amount of employees of the two software organizations can be interviewed. In order to mitigate this, we collaborated with employees engaged in a wide spectrum of activities, roles and different tools, so to ensure that the sample attained a high degree of representativeness of the situation.

Last but not least, the reliability of the results, defines the measure of which the study can be reproduced in the future from different researchers and still produce the same results. This is determined by the extent to which the collected data and their subsequent analysis are dependent the original researchers. A typical scenario of this threat, includes cases where the data collection or analysis process are not illustrated enough, therefore, they are cumbersome to reproduce by third persons.

5 Results

During the case studies, we conducted six semi-structured interviews, based on an essentially similar interview guide. The interviews were recorded and later transcribed. The data found below, consist our effort to structure and visualize them, using tables, so to facilitate the correlation between the different variables and the extraction of findings. Each table is accompanied by a description in the appropriate subsection. The results have been divided into sections, grouped by the research question they intend to answer.

Additionally, a synopsis of the specific elements, suggested made by the interviewees, can be found in table 14 in the Appendix. This table is of particular interest because it displays the entire latitude of the various benefits, challenges and attributes of shared tools, according to the employees of the two large software organizations that were investigated. It will be used in the following sections in order to designate the most significant results.

5.1 Background

Table 5.1 contains the background information for each interview. They illustrate the specific context for each interview and were gathered at the beginning of each discussion. This was done, not only to acclimatize the interviewees but also offer a broader perspective, for the data that would follow to be placed into. They are particularly useful, in order to quickly establish an overview of the specific setting for each case, for example regarding the product that is being developed, or whether the interviewees actively develop or merely use a shared tool or toolchain. These results are not directly mapped to a specific research question, however, they can facilitate discussion by designating the given context on each occasion.

5.2 Benefits

Table 5.2 eloquently illustrates the various benefits of share tools, structured according to the BAPO framework. What is more, the benefits on each dimension are also weighted, depending on the importance the interviewees would designated them as. For example, the first row of the table, can be interpreted as interviewee 1, in company A does not believe there are particularly many benefits of using shared tools in the product itself, however firmly supports that they are advantageous to the process.

Generally, the subjects argued that by sharing tools, the total productivity is increased, as the shared knowledge is mainly enhanced from the organization and process perspectives. This allows tasks to be completed faster. This observation was confirmed by most of the interviewees from both companies as it is shown in the table 5.2. Additionally, shared tools seem to facilitate the reuse of software artifacts, while improving the communication and collaboration within the organization. Some characteristic excerpts from the discussions follow, which can give the reader a better idea of the collected information regarding the benefits of shared tools and toolchains. For a complete outlook on the benefits suggested during the case study, please refer to table 14 in the Appendix.

An employee from company B argued on the topic of the advantages offered by shared tools:

> "The tools are setup to work with a specific organizational structure and the process we chose is based on certain team size. In that sense, it works well that we share both the tools and the process."

An interviewee, from company B, claimed in regards to the benefits of shared toolchains:

Table 1. Context information

ID	Organization	Role	Product	Develop & share tool(chain)	Use shared tool(chain)
1	B	Software architect	CI	TRUE	TRUE
2	B	Web developer	CI	TRUE	TRUE
3	B	Software developer	BS	FALSE	TRUE
4	B	Software developer	BS	FALSE	TRUE
5	A	Software developer	CI	TRUE	TRUE
6	A	Software developer	ECU	TRUE	TRUE

*CI = Continuous Integration system, ECU = Engine Control Unit software, BS = Base Station software

Table 2. Benefits

ID	Product	Process	Business	Organization	Architecture
1	4	1	3	2	5
2	1	NA	NA	2	3
3	3	2	NA	1	NA
4	NA	3	1	2	NA
5	2	1	1	3	5
6	4	1	3	2	3

*1 = most significant, 5 = least significant NA= Not Applicable

Table 3. Challenging aspects

ID	Product	Process	Business	Organization	Architecture
1	1	2	NA	3	NA
2	3	2	NA	NA	1
3	NA	2	NA	1	NA
4	2	1	NA	3	NA
5	2	3	1	4	NA
6	NA	NA	NA	1	2

*1 = most significant, 5 = least significant NA= Not Applicable

> ”Having shared toolchain is definitely a benefit for the knowledge management perspective in organizations.”

Additionally, a different individual from the same company, noted in regards to the advantages gained by the process:

> ”Since we are using the same tools as the whole of the organization, everything is setup, so process wise is easier to start with.”

As to the benefits on the business aspects of the organization, the majority of the interviewees agreed that they believe they can clearly determine benefits. A person from company A told the researchers:

> ”It will save money by not having too much customized solutions.”

Several interviewees could foresee benefits to the product itself, by the use of shared tools and toolchains. Particularly, a characteristic example, from an interview with individuals of company A, regarding benefits to the product includes the following:

> ”Sharing tools means also you can spend more resources on them as it will benefit the product in that sense.”

5.3 Challenges

The gathered results regarding the various challenges involved in the use of shared tools, are illustrated in the following sections. They are divided into the challenging aspects of software development, according to the BAPO framework and then, a list of various common challenges, their significance according to their interviewees and whether there has been a devised mitigation strategy. In table 14 the whole spectrum of challenges that were mentioned during the interviews can be seen.

5.3.1 Challenging aspects

Data collected on challenging aspects, are summed up in table 5.3.1 and are organized in the same manner as the benefits, in table 5.2, as illustrated in section 5.2. The challenges, grouped by BAPO framework's dimensions and designated with their respective weights, give us an outlook on the general situation, from the perspective of the interviewees, in regards to challenges. Specifically, when it comes to the challenges, the most noticeable and distinct one revolves around the process, as table 5.3.1 suggests.

A strong opinion regarding obstacles due to shared tools, was expressed by an employee of company B:

> ”Sometimes developers need to change between the tools too often, especially in cross functional teams. It's quite a waste of time from the process perspective.”

This seems to be in accordance with the interviewees general perception that knowledge management can be problematic when too many shared tools are involved. Sim-

Table 4. Challenge / Mitigation strategy

ID	Intellectual Property	Ownership	Versioning	Evolution
1	5 / FALSE	1 / TRUE	5 / FALSE	2 / TRUE
2	5 / FALSE	2 / FALSE	1 / TRUE	3 / FALSE
3	2 / TRUE	3 / TRUE	3 / TRUE	3 / TRUE
4	3 / TRUE	3 / TRUE	1 / TRUE	3 / TRUE
5	2 / TRUE	1 / FALSE	2 / TRUE	2 / TRUE
6	3 / TRUE	5 / TRUE	3 / TRUE	3 / TRUE

*1 = most significant, 5 = least significant
TRUE = existing mature mitigation strategy
FALSE = NOT existing mature mitigation strategy

ilarly, a person from company A mentioned additional challenges in regards to this aspect, of having an abundance of tools in use:

> "Having too many shared tools, you will get knowledge management problems when products going to the market. The maintainability of the product might be affected if you have too many tools, since the problem of maintaining the knowledge of how to use the tools will be larger."

Aside from the process and product perspective, the challenges to the organization can not be neglected as well, with 5 out of 6 interviewees mentioned the problem. A person from company B claimed:

> "Sometimes the tools have assumptions about the team structure. And also when people are leaving, the responsibility for the tools changes as well. Everything becomes very rigid when you have tools.

At the same time, an employee from company A supported the opinion that tools might even hinder innovation:

> Having too strict policies for introducing shared tools might be demotivating certain developers and it might also slow down the innovation."

The same participant argued that since sharing and reusing tools can be beneficial, a way to tackle the above consequence would be to allow the developers to introduce new tools, if the need arises.

5.3.2 Challenging factors and mitigation strategy

Table 4 illustrates the interviewee's responses to various common challenges in the specific context, as well as the existence of a mitigation strategy against these concerns. During the interviews, it became apparent the ownership and responsibility of shared tools, is a considerable challenge to most of the individuals we discussed with.

Yet, despite this problem being somewhat common, some of them are encountering difficulties mitigating or finding a solution about it. When investigating a bit deeper into the issue, we identified a divergence between the users and the developers of shared tools. Particularly, in an occasion from company B, the user takes as granted that:

> "Usually, there should be a team that supports tools"

On the other hand, a developer of shared tools, from the same company, discussed that in their work, they are specifically careful with taking the responsibility of a tool.

> "Owning a tool is expensive, and it requires a lot of effort."

Generally, owning a tool and thus accepting its responsibility was one of the most mentioned challenges of shared tools. The evolution of shared tools is another challenging factor which was considered of high importance among the other issues that exist. The same developer from company B argued on the matter:

> Most of the developers have some legacy tools or APIs to get rid of, yet they have to support them, porting them and deprecating them gradually".

What is more to the evolution management, many of the participants argued that a good versioning system has to be devised. One example that came up involved different versions of the same tool for separate clients. Additionally, it is worth to mention that some of the participants found defining the appropriateness of a shared tool a challenging factor, something that we also came across in previous studies.

5.4 Shared tool characteristics

To define a set of suggested characteristics for shared tools and toolchains, we first present the results regarding the development phases which according to the interviewees are relevant to the use of shared tools. This is followed by, the various ideas collected from the case study participants, on the different attributes they strive for. The last table includes the findings on how the employees evaluate the significance of numerous common shared tool aspects.

5.4.1 Shared tools suitability

In the effort of the researchers to determine which development phase the tools are mostly appropriate for, in order to determine the various generic properties a shared tool should have, the participants in the interviews were asked to designate which software development phase, the shared tools mostly fit in. The summary of the responses is displayed in table 5.4.1.

The results, designate the integration phase to be of high significance as to where shared tools are suitable. Other

Table 5. Shared tools suitability

Software development activities	Positive responses
Implementation	3
Integration	6
Testing	3
Deployment	4

than that, the rest of the stages enjoy similar preference by the interviewees. Specifically for the implementation, an employee of company B mentioned:

> "When it comes to implementation, I don't think that organization should force everybody to work with the same tools."

This could imply a strict workflow, introduced by hard-to-use tools, can also pose obstacles to development and creativity.

5.4.2 Characteristics for sharing

The interviewees were asked to share their thoughts on the characteristics a tool or a toolchain should have, in order facilitate its sharing across a software organization. The rightmost column of table 14 summarizes the characteristics suggested during the discussions, that shared tools should attain, as well as the frequency they were mentioned by the interviewees.

There was a general consensus on the need for active support on the tools. This was mentioned by both the developers and the ones who merely use shared tools. This support, implying responsibility and ownership was admittedly an intricate task, as it was previously discussed and illustrated in section 5.3.2. Extensive and understandable documentation was another popular choice among the individuals that participated in the interviews.

Additionally, some of the participants required the tools to be usable, interoperable, attributes that were also mentioned in the literature and will facilitate the combination of different tools to form toolchains. Last but not least, some of the interviewees suggested tools need to be tested and dependable, conditions that can be traced back to our findings from the related literature, which indicated the lack of trust developers commonly exhibit towards tools.

5.4.3 Other attributes

Lastly, the interview participants were asked to express their opinion on various common characteristics of shared tools and to define their significance from their own perspective. An attempt to collect and visualize the results can be seen on table 6. The table follows the format that was previously explained, with each attribute being assigned to a measurement of significance, according to each interviewee.

Among the available attributes, infrastructure, standardization and openness are the most highly evaluated ones. Particularly, the shared tool developers drew the picture that it is important to have a communication channel to discuss with the users. This could be a web forum or a physical meet-up. On the other hand, from a user perspective, they appreciate a team of experts for troubleshooting and help, with high availability. Additionally, there was a preference towards web-based tools, as opposed to locally installed ones, possibly due to them being easier to acquire and install.

Several interviewees, considered standardization and openness as interrelated factors. These attributed were highly esteemed. Additionally, it was mentioned that standardization was a precondition of openness. A participant from company B, mentioned in this regard:

> "If you have openness you have to have standardization, otherwise it doesn't work."

On the matter, an interviewee from company A claimed:

> "When I want to integrate tools that I want in a larger environment or ecosystem, working with common data formats makes a lot of things much easier. So openness and standardized APIs also depend on the quality and maturity of the tools."

5.5 Assessment method

There were no specific case study results, exclusively used in order to facilitate the formulation of a shared tools assessment method. That being said, elements from all the previous research questions, including the background information on the participants and the different contexts, were utilized in order to initiate the discussion. Here, it is noteworthy that in two occasions there was a reference to selecting the appropriate tools being a challenging task.

6 Discussion

The empirical data from the case study, combined with the bibliographic evidence from the literature review enabled the researchers to formulate answers to the research questions and reach several conclusions related to the subject of this study. The data from the two sources, i.e. the literature review and the case study, will be utilized in order to define the challenges and benefits of shared tools in software organizations, suggestions on how to share them and an early approach on a shared tool assessment method. The answers to the research questions will be formulated based on the significance of the findings from each source, i.e. the

Table 6. Other attributes

ID	Infrastructure	Community	Architecture	Accessibility	Openness	Standardization
1	2	3	5	2	1	1
2	1	4	5	2	4	1
3	1	2	5	4	3	1
4	5	5	5	3	2	5
5	1	1	5	4	1	2
6	2	3	5	2	2	2

*1 = most significant, 5 = least significant

case study and the literature review, as well as whether we can synthesize data from the two sources.

6.1 Benefits of shared tools in software organizations

As to the benefits of shared tools and toolchains in software organizations, results from the case study and the literature review appear to be to a considerable extent in alignment. Generally, it can be observed that the members of the software organizations who were interviewed identify shared tools to be mostly beneficial to the process, organizational and business aspects. This means that the development process was improved by the introduction of shared tools and toolchains, the organizational structure becomes clearer and seamless as well as the profitability of the products appears to be increased.

Particularly, based on the case study interviews, there seems to be a distinctly positive effect, on the productivity. This was in alignment with findings from the bibliographic research, as presented in section 2.2. Tasks appear to be completed faster, as a consequence of the adoption of shared tools and toolchains. In the meanwhile, costs are decreased, which in turn either increases the competitiveness of the final product or enables the organization to invest more in it, without additional resources.

Furthermore, the decrease in development time is often the consequence of automation in one of the development phases, such as testing or integration. This automated process can benefit the quality of the final product. An example could involve shared testing tools, increasing the test coverage, as opposed to manual ones. Moreover, the use of shared tools, can catalyze and augment the advantages of model driven engineering, by facilitating the various model transformations and handling with reusable tools, across the organization. This, in turn, benefits the communication among the different teams.

6.2 Challenges of shared tools in software organizations

The utilization of shared tools in software organizations is accompanied by a plethora of challenges. They were prevalent in both the findings from the case study, as well as the bibliographic research. The various difficulties, seem to be hindering mainly the development process aspects, while it should be noted most of the interview participants claimed lack of opinion, regarding challenges as seen from a business perspective. This could be attributed to the fact, that none of the interviewees held a position very high in management.

Additionally, there appears to be some convergence on the various issues that are faced, between the case study and the literature. Particularly, the evolution management, in a shared tool environment within an organization, can be difficult. The same can be argued from a software ecosystem perspective, regarding the various architectural challenges that commonly exist. Next, a persistent impediment, according to various sources in the literature regarding the use of tools and toolchains, is the lack of interoperability. Interoperability, on the other hand, was considered as a very important characteristic by the interviewees for tools, however, was not specifically mentioned as a challenging factor.

Many of the case study participants, among both of the two software organizations, found ownership and responsibility for a tool, to be a particularly troubling factor. It was admitted that, the fact that the implied responsibility to support a shared tool after releasing it, was a discouraging agent. Furthermore, another point of probable alignment between the literature review and the case study findings could be the high cost of creating a shared tool and a toolchain. This, due to the fact that it is a manual procedure. During the discussions with the employees in the software organizations, it was mentioned that it is often cumbersome to learn many different tools, as well as maintaining the relevant knowledge on an organizational level, while also shared tools can be a limiting factor in some occasions. These two observations, from the different sources, can be

plausibly correlated. The high cost and complexity of developing tools and toolchains can justifiably be complemented by the observation that it is usually difficult to learn and use shared tools.

Next, the need for openness and an assessment method to specify the suitability were noted on multiple occasions as two typical important challenges, accompanying the use of tools and toolchains. These aspects were encountered both in the literature and during the interviews with the developers. Moreover, versioning was frequently mentioned as an important cause of problems during the case study, therefore, it should be considered as a factor that often poses challenges in the adoption and development of shared tools.

6.3 Suggested characteristics of shared tools

During the case study and the literature review, we came across a plethora of useful characteristics that tools shared within software organizations should attain. One of those attributes, as designated by both the past research on the subject as well as the interviewees, is interoperability. The ability of a tool, to be easily interfaced with other, was one of the substantial attributes of a shared tool.

Additionally, this "virtue" seems to be interrelated with some other suggested characteristics, such as functionality, usability, openness and standardized interfaces. Particularly, a tool that is able to be easily combined with other tools or the rest of the system makes it more trivial to use and increases its learnability. These, in turn, go hand in hand with the proposed openness features by both the literature and the case study, such as the import and export functions or the standardized communication protocol.

Furthermore, some of the participants in the case study argued that they would appreciate better support and documentation, something that should decrease the learning curve. Therefore, with a better overall support both in terms of training material, e.g. usage manuals, Wikis, documentation pages and human capital, i.e. support teams, the effort and cost for developers to familiarize themselves with tools and toolchains should be decreased. Finally, we observed a trend among the participants to highly value web-based tools, which could imply that they are easier to get acquainted with and distribute.

6.4 An approach towards a shared tool assessment method

Building upon the previous discussion and particularly the findings on the benefits, the challenges as well as the proposed characteristics of shared tools, there was observed a high degree of alignment between the various aspects of shared tools with the different attributes suggested by the criteria based assessment. The criteria based assessment, described in section 4.3, is a generic software evaluation framework, that enables assessments judging on the usability, maintainability and sustainability of software artifacts.

In order to customize the various criteria, to the specific circumstances of each context, that a shared tool or toolchain are evaluated within, specific weights should be assigned to each criterion, so to compensate for the various domain specific requirements or use case. Moreover, the need to deepen the shared tools assessment method and adapt it to the appropriate circumstances might arise. In that occasion, the evaluation could be complemented with relevant questions from the tutorial based assessment.

7 Validity threats

Threats to validity to case studies that intend to extract qualitative data have always been a subject of concern to the research community. This, due to the fact that the persons conducting the study have to be at certain points subjective and utilize their own personal perspective in order to respond creatively and rigorously [25] to the circumstances that are being developed. In order to identify, organize and therefore be able to mitigate the various risk factors, we outlined the various aspects of validity in Section 4.4. Specifically, the classification scheme by Runeson et. al. (2009) [17] was used in order to identify the various threats among the various aspects of this study. In the next sections, we shall outline the various mitigation tactics that were devised in order to assure the various aspects of validity.

7.1 Construct validity

Construct validity can be briefly described as the extent to which the chosen research methodology corresponds with the concepts and theories that are being investigated. A first step in order to assure the integrity of construct validity was to establish a common baseline with the interviewees, regarding the common key concepts at the beginning of each interview. Each participant was asked to give a definition of the various terms that are discussed, in order to ensure the conceptual alignment between the two sides.

Additionally, the interview guide was validated by our two supervisors for academic suitability and continuously improved for domain appropriateness. Particularly, the supervisors helped to ensure that the questionnaire stood by high research standards and was appropriate for the context. Moreover, the pilot interview at company A, as well as the various minor adjustments that were made, enabled us to use questions and terms understandable by the interviewees and relevant to their domain's technical vocabulary.

7.2 Internal validity

Internal validity sums up to the existence of factors unknown to the persons conducting the research that can affect the data collection process. We tried to minimize risks towards the internal validity of our study, by selecting participants from different teams or companies, tasked with distinct responsibilities. This enabled us to triangulate the results and minimize the effect of unprecedented factors to the total results of this research.

7.3 External validity

The external validity can be attributed as the degree to which the results of a study can be generalized and them to be usable by future researchers or other individuals. In other words, whether the results are applicable in related but separate contexts. This is recognized as one of the most major risks of the study, as the number of individuals and tools from the two software organizations that were investigated in the current research are merely a fraction of the total number of developers and tools utilized in those companies.

That being said, the fact that the participants that were selected interviewee profiles differed greatly, we attempted to mitigate risks to the external validity and allow our results to be feasibly generalizable, to a certain extent. Additionally, the software organizations that were selected, are engaged in two very broad technological domains, namely the automotive and telecommunications and are committed in an attempt to improve their processes. It is plausible to argue that the various phenomena encountered in those two domains, should be relevant to different ones as well.

On the other hand, both of the organizations that were investigated, are considered large. Subsequently, the applicability of the results to medium and small companies is questionable and calls for further research.

7.4 Reliability

The reliability of the study can be characterized as the extent to which the researchers conducting it, are emphatically affecting its outcome, in a way that it ceases to be reproducible by different researchers, following the same methodology. In other words, reliability is determined by whether future researchers following the same process, are able to extract the same findings. It should be noted that there are some concerns, as to this aspect, due to the fact that a relatively small number of interviews was conducted. Additionally, the process followed during the literature review was not systematic, therefore not the same bibliographic results will be necessarily be discovered.

In order to increase the reliability of the study, the data collection process was conducted in two different software organizations, including participants with distinct roles and responsibilities. Next, during the interviews we tried to not affect the participants with previous knowledge, e.g. common challenges in the particular context, that we had previously accumulated during the case study. This, to ensure the collected data would be undiluted data, free from bias.

8 Conclusion

During this research, a literature review and a case study were conducted, in order to further investigate shared tools and toolchains, as well as their various implication in software organizations. The findings were formulated based upon the synthesis of results from the relevant bibliography, along with the analysis of collected data from a case study, involving employees from two major software organizations, in Gothenburg, Sweden.

The results indicate that common benefits and challenges of tools in the literature are also encountered by the participants of the two software companies that were investigated, such as the increase of productivity and the lack of interoperability respectively. Their ability to increase the degree of automation, allow for faster implementation and higher quality products. Furthermore, the use of shared tools and toolchains enables reaping the benefits of model driven engineering. Additionally, sharing tools facilitate cross-organizational communication.

On the other hand, managing the evolution and versioning of shared tools can be challenging tasks, something that is encountered in the literature for artifacts within software ecosystems as well. Then, another important issue that the shared tool developers often find themselves faced with, is the ownership and responsibility. The fact that it could be implied that they will have to support a tool once they share it, can be hard to afford and requires considerable effort. In addition to this, learning new tools is often hard and so is their development, which both translate to high cost.

Generally, using shared tools, appeared to positively contribute to the process, organization and business aspect of the software development. At the same time, their adoption can often lead to problems, mainly regarding some other process aspects of software development. This high level information on where benefits and challenges are mostly experienced, can serve as a directive to managers who wish to improve the certain dimensions of their organizations or are trying to narrow down the source of various impediments.

Some distinct and novelty characteristics of shared tools were surfaced through the case study, such as the need for extensive and elaborate documentation, as well as dedicated user groups for support. Such resources could lower the

learning curve, therefore, decrease their cost. Next, a major attribute that tools should be represented by is interoperability. Interoperability can, in turn, be correlated with other traits, such as usability, openness and standardized interfaces. These qualities allow shared tools to be combined and reused easier, which is typically essential to their purpose.

Additionally, the appropriateness of a generic software assessment method was verified to be applicable for shared tools and toolchains. Specifically, building upon the previous implications of shared tools in software organizations, a high degree of alignment was observed between the various aspects of shared tools and those evaluated by the criteria based assessment framework. This lead us to claim that one can apply the criteria based assessment, which reviews software artifacts in terms of usability, maintainability and sustainability, on shared tools and toolchains.

During the data analysis, it was recognized that a larger amount of first-degree empirical data is necessary in order to acquire a deeper insight of the various specific challenges and benefits of each software organization. Therefore, it is suggested that future research is oriented towards refining the proposed assessment method, by utilizing more data points. What is more, it would be of particular interest to attempt to generalize the findings and place them in a broader perspective. Specifically, to investigate the implications of shared tools and toolchains, within a software ecosystem that involves external actors.

References

[1] Hanssen, G. K. (2012). A longitudinal case study of an emerging software ecosystem: Implications for practice and theory. Journal of Systems and Software, 85(7), 1455-1466.

[2] van den Berk, I., Jansen, S., & Luinenburg, L. (2010, August). Software ecosystems: a software ecosystem strategy assessment model. In Proceedings of the Fourth European Conference on Software Architecture: Companion Volume (pp. 127-134). ACM.

[3] Knauss, E., & Hammouda, I. (2014, August). EAM: Ecosystemability assessment method. In 2014 IEEE 22nd International Requirements Engineering Conference (RE) (pp. 319-320). IEEE.

[4] Hammouda, I., Knauss, E., & Costantini, L. (2015, May). Continuous API design for software ecosystems. In Rapid Continuous Software Engineering (RCoSE), 2015 IEEE/ACM 2nd International Workshop on (pp. 30-33). IEEE.

[5] Cao, Q., & Gong, Z. (2015). API Design Considerations: An Empirical Assessment Approach. University of Gothenburg. Retrieved from http://hdl.handle.net/2077/39979

[6] Sekitoleko, N., Knauss, E., Damian, D., Hammouda., I., & Lantz, J. (2015). The Role of Secondary Software in Automotive Ecosystems. In Proceedings of the European Open Symposium on Empirical Software Engineering, Lille, France.

[7] Bosch, J. (2010, August). Architecture challenges for software ecosystems. In Proceedings of the Fourth European Conference on Software Architecture: Companion Volume (pp. 93-95). ACM.

[8] Barth, M., Drath, R., Fay, A., Zimmer, F., & Eckert, K. (2012, September). Evaluation of the openness of automation tools for interoperability in engineering tool chains. In Emerging Technologies & Factory Automation (ETFA), 2012 IEEE 17th Conference on (pp. 1-8). IEEE.

[9] Schlegel, C., Steck, A., Brugali, D., & Knoll, A. (2010). Design abstraction and processes in robotics: from code-driven to model-driven engineering. In Simulation, Modeling, and Programming for Autonomous Robots (pp. 324-335). Springer Berlin Heidelberg.

[10] Biehl, M., & Torngren, M. (2012, August). A cost-efficiency model for tool chains. In Global Software Engineering Workshops (ICGSEW), 2012 IEEE Seventh International Conference on (pp. 6-11). IEEE.

[11] Biehl, M., & Torngren, M. (2012, July). An Estimation Model for the Savings Achievable by Tool Chains. In Computer Software and Applications Conference Workshops (COMPSACW), 2012 IEEE 36th Annual (pp. 488-492). IEEE.

[12] Biehl, M. (2013). A modeling language for the description and development of tool chains for embedded systems.

[13] Porter, J., Volgyesi, P., Kottenstette, N., Nine, H., Karsai, G., & Sztipanovits, J. (2009, June). An experimental model-based rapid prototyping environment for high-confidence embedded software. In Rapid System Prototyping, 2009. RSP'09. IEEE/IFIP International Symposium on (pp. 3-10). IEEE.

[14] Walderhaug, S. (2013). Design and evaluation of the ModelHealth toolchain for continuity of care web services. Automated Software Engineering, 20(2), 185-235.

[15] Wildmoser, M., Philipps, J., & Slotosch, O. (2012). Determining Potential Errors in Tool Chains. In Computer Safety, Reliability, and Security (pp. 317-327). Springer Berlin Heidelberg.

[16] Van Der Linden, F., Bosch, J., Kamsties, E., Känsälä, K., & Obbink, H. (2004). Software product family evaluation. In Software Product Lines (pp. 110-129). Springer Berlin Heidelberg.

[17] Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. Empirical software engineering, 14(2), pp. 47-55 and 131-164.

[18] Waszecki, P., Lukasiewycz, M., Masrur, A., & Chakraborty, S. (2013). How to engineer tool-chains for automotive E/E architectures? ACM SIGBED Review, 10(4), 6-15.

[19] Karsai, G., Biswas, G., Abdelwahed, S., Mahadevan, N., & Manders, E. (2006). Model-based software tools for integrated vehicle health management. Paper presented at the , 2006 8 pp.-442.

[20] Kern, H., & Kühne, S. (2009, June). Integration of microsoft visio and eclipse modeling framework using m3-level-based bridges. In Proceedings of Second Workshop on Model-Driven Tool and Process Integration (MDTPI) at ECMFA, CTIT Workshop Proceedings (pp. 13-24).

[21] Yin, R. K. (1994). Case study research: Design and methods (2.th ed.). Thousand Oaks, CA: Sage.

[22] Höst, M., Runeson, P., Institutionen för datavetenskap, Faculty of Engineering, L., Lunds universitet, Department of Computer Sciences. . Departments at LTH. (2007). Checklists for software engineering case study research.

[23] Jackson, M., Crouch, S., & Baxter, R. (2011). Software evaluation: criteria-based assessment. Software Sustainability Institute, The University of Edinburgh. Retrieved from http://goo.gl/Ah7B1w

[24] Jackson, M., Crouch, C., & Baxter, R. (2011). Software Evaluation: Tutorial-based Assessment. Software Sustainability Institute Guides, The University of Edinburgh. Retrieved from http://goo.gl/JTdw5J

[25] Maxwell, J. A. (1992). Understanding and validity in qualitative research. Harvard Educational Review, 62(3), 279.

[26] Burden, H., Heldal, R., & Whittle, J. (2014, September). Comparing and contrasting model-driven engineering at three large companies. In Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (p. 14). ACM.

[27] Pohjolainen, P. (2002). Software Testing Tools. Department of Computer Science and applied mathematics, University of Kuopio. Retrieved from http://goo.gl/RxQe7C

[28] Sendall, S., & Kozaczynski, W. (2003). Model transformation the heart and soul of model-driven software development (No. LGL-REPORT-2003-007).

[29] Bruyninckx, H., Klotzbücher, M., Hochgeschwender, N., Kraetzschmar, G., Gherardi, L., & Brugali, D. (2013). The BRICS component model: A model-based development paradigm for complex robotics software systems. Paper presented at the 1758-1764.

[30] Mustafa, K. M., Al-Qutaish, R. E., & Muhairat, M. I. (2009, December). Classification of software testing tools based on the software testing methods. In 2009 second International Conference on Computer and Electrical Engineering (pp. 229-233). IEEE.

[31] Seth, N., & Khare, R. (2015, December). ACI (automated Continuous Integration) using Jenkins: Key for successful embedded Software development. In 2015 2nd International Conference on Recent Advances in Engineering & Computational Sciences (RAECS) (pp. 1-6). IEEE.

[32] Chen, L. (2015). Continuous delivery: Huge benefits, but challenges too. IEEE Software, 32(2), 50-54.

[33] Schultis, K. B., Elsner, C., & Lohmann, D. (2014, November). Architecture challenges for internal software ecosystems: a large-scale industry case study. In Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (pp. 542-552). ACM.

[34] Voget, S., & Favrais, P. (2010). How the concepts of the Automotive standard 'AUTOSAR' are realized in new seamless tool-chains. In 2010 European congress of Embedded Real-Time Software and Systems, Toulouse.

[35] Trochim, W. M. (2006). Types of surveys. Retrieved from http://goo.gl/VfPZij

[36] Whittle, J., Hutchinson, J., Rouncefield, M., Burden, H., & Heldal, R. (2013). Industrial adoption of model-driven engineering: Are the tools really the problem?. In Model-Driven Engineering Languages and Systems (pp. 1-17). Springer Berlin Heidelberg.

[37] Smith, E. K., Bird, C., & Zimmermann, T. (2015, May). Build it yourself! homegrown tools in a large software company. In Software Engineering (ICSE), 2015 IEEE/ACM 37th IEEE International Conference on (Vol. 1, pp. 369-379). IEEE.

[38] Wolvers, R., & Seceleanu, T. (2013, September). Embedded Systems Design Flows: Integrating Requirements Authoring and Design Tools. In Software Engineering and Advanced Applications (SEAA), 2013 39th EUROMICRO Conference on (pp. 244-251). IEEE.

A Appendix

A.1 Interview Guide

1. What is your role in the company?
2. What is your definition of a shared tool/toolchain?
3. Do you have any existing shared tool/toolchain in your organization? If no, are you planning one?
4. Could you please discuss the benefits of shared tools with respect to product, process, business, organization, and architecture dimensions?
5. Can you rank the importance of those dimensions with respect to the benefits of shared tools?
6. Could you please discuss the challenges of shared tools with respect to product, process, business, organization, and architecture dimensions?
7. Can you rank the importance of those dimensions with respect to the challenges of shared tools?
8. If not covered, how do you rate the importance of those factors with regard to challenges:
    - Intellectual property
    - Ownership/responsibility
    - Versioning
    - Evolution
9. If not already covered, how do you address those challenges?
10. In what software development activities shared tools fit best? E.g. Implementation, Integration, Deployment
11. How tools should be shared in the organization?
12. What characteristics should a tool have for sharing in the organization?
13. How do you see the importance of those factors for a shared tool?
    - Infrastructure (e.g. forum, etc)
    - Community (inner source versus open source)
    - Architectural issues (such as object orientation)
    - Accessibility (e.g. local installation versus web-based)
    - Openness (e.g. import/export features)
    - Standardization (e.g. well-defined data exchange format)

A.2 Interview data

The following tables include a summary of the interviews, classifying the findings into three categories:

- Benefits of shared tools
- Challenges of shared tools
- Suggested characteristics or attributes of shared tools

Table 14 visualizes the information gathered from all the resources. If elements were discovered in more than one interview, their frequency was included next to them, inside parentheses. Additionally, table 7 displays the sequence the various interviews were conducted chronologically, with the first two interviews being conducted with employees of company A and the rest with participants from company B.

Table 7. Interview chronological data

ID	Date
6	November 2015
5	February 2016
3	March 2016
4	April 2016
1	May 2016
2	May 2016

Table 8. Interview #1

Benefits	Challenges	Attributes
Easier to setup and get started	Can be a limiting factor	Interoperability
Cost decrease	High implementation cost	Few dependencies
Maintainability	Assumptions on organizational structure	User group for support
Seamless organization	Require responsibility	Communication infrastructure
Enforce a common architecture	Require good versioning	Standardized interfaces
		Openness
		Web-based

Table 9. Interview #2

Benefits	Challenges	Attributes
Facilitates reuse	Require robustness	Solid infrastructure
Seamless organization	Ad-hoc share or usage	Web-based
Sustainability	Misuse	Standardized interfaces
Performance	Need to support legacy code or APIs	Consistency
	Require good versioning	Interoperability

Table 10. Interview #3

Benefits	Challenges	Attributes
Robustness	Difficult to learn	Documentation
Efficiency	Knowledge management (too many tools)	Wiki
Consistent communication	Hard to assess the tool appropriateness	User group for support
		Easy to distribute

Table 11. Interview #4

Benefits	Challenges	Attributes
Cost decrease	Difficult to learn	Tested
Knowledge management (reduced number of tools)	Knowledge management (too many tools)	Dependable
	Require good versioning	Documentation
		User group for support
		Usability
		Openness

Table 12. Interview #5

Benefits	Challenges	Attributes
Knowledge management (reduced number of tools)	Can be a limiting factor	Usability
Maintainability	Can hinder innovation	Documentation
Facilitate reuse	Knowledge management (too many tools)	Up-to-date
Increased quality	Low maintainability (too many tools)	Openness
	Lack of certification	Standardized interfaces
	Hard to assess the tool appropriateness	User group for support
	Require responsibility	

Table 13. Interview #6

Benefits	Challenges	Attributes
Faster production	Ad-hoc approach	Web-based
Better communication	No top-down implementation	Documentation
Facilitate testing	Employee resistance	Discoverability
Enforce a common architecture	Narrow domain perspective	Relevance to domain
Common vocabulary	Competing and conflicting tools	Tested
Facilitate cross-organizational model handling	Bad documentation	Ranked
	Require responsibility	Standardized interfaces
	Require feedback	Easy to setup

Table 14. Interviews summary

Benefits	Challenges	Attributes
Knowledge management(2) (reduced number of tools)	Require responsibility(3)	Documentation(5)
Cost decrease(2)	Require good versioning(3)	User group for support(4)
Maintainability(2)	Knowledge management(3) (too many tools)	Standardized interfaces(4)
Seamless organization(2)	Can be a limiting factor(2)	Openness(3)
Enforce a common architecture(2)	Difficult to learn(2)	Web-based(3)
Facilitate cross-organizational model handling	Hard to assess the tool appropriateness(2)	Tested(2)
Facilitates reuse(2)	High implementation cost	Interoperability(2)
Easier to setup and get started	Assumptions on organizational structure	Usability(2)
Sustainability	Requires robustness	Few dependencies
Performance	Misuse	Communication infrastructure
Robustness	Need to support legacy code or APIs	Solid infrastructure
Efficiency	Can hinder innovation	Consistency
Faster production	Low maintainability (too many tools)	Wiki
Consistent communication	Lack of certification	Easy to distribute
Better communication	Ad-hoc approach	Dependable
Common vocabulary	No top-down implementation	Up-to-date
Increased quality	Employee resistance	Discoverability
Facilitate testing	Narrow domain perspective	Relevance to domain
	Competing and conflicting tools	Ranked
	Bad documentation	Easy to setup
	Require feedback	