The Importance of Architectural Knowledge in Integrating Open Source Software

Klaas-Jan Stol, Muhammad Ali Babar, Paris Avgeriou

The Importance of Architectural Knowledge in Integrating Open Source Software

Klaas-Jan Stol[1], Muhammad Ali Babar[2], and Paris Avgeriou[3]
[1]Lero—The Irish Software Engineering Research Centre
University of Limerick, Ireland
[2]IT University of Copenhagen, Denmark
[3]University of Groningen, the Netherlands
klaas-jan.stol@lero.ie, malibaba@itu.dk, paris@cs.rug.nl

Abstract. Open Source Software (OSS) is increasingly used in Component-Based Software Development (CBSD) of large software systems. An important issue in CBSD is selection of suitable components. Various OSS selection methods have been proposed, but most of them do not consider the software architecture aspects of OSS products. The Software Architecture (SA) research community refers to a product's architectural information, such as design decisions and underlying rationale, and used architecture patterns, as Architecture Knowledge (AK). In order to investigate the importance of AK of OSS components in integration, we conducted an exploratory empirical study. Based on in-depth interviews with 12 IT professionals, this paper presents insights into the following questions: 1) what AK of OSS is needed? 2) Why is AK of OSS needed? 3) Is AK of OSS generally available? And 4) what is the relative importance of AK? Based on these new insights, we provide a research agenda to further the research field of software architecture in OSS.

Keywords. Open Source Software integration, component-based development, architectural knowledge, software architecture, OSS Integrator, survey

1 Introduction

Software development organizations can adopt Open Source Software (OSS) in various ways [1]. One way is to integrate OSS components in Component-Based Software Development (CBSD), which has become a standard way of building large-scale systems. In such a setting, a software development organization has the role of *OSS integrator* [2]. Building systems from externally and independently developed components is not without risks. Garlan et al. reported their experiences of using only four components and experienced significant integration problems. They identified the root cause of their problems to be *architectural mismatch* [3]. The increasing use of OSS components as an alternative to Commercial Off-The-Shelf (COTS) components in CBSD [4] can result in more architectural mismatch

challenges for OSS integrators. Despite the common concerns about OSS licenses and a lack of support and documentation, the use of OSS offers various benefits such as access to the source code and low costs [5], which means that industry is likely to continue to use OSS products.

Various authors have reported studies of CBSD with OSS products [6-9], or more generally with Off-The-Shelf (OTS) components (both OSS and COTS) [10, 11]. Most of these studies focus on evaluation and selection of components, which is a critical aspect in CBSD. While software architecture (SA) has long been recognized as an important success factor in CBSD [12], little attention has been paid so far to architectural aspects of OSS components, such as a component's architectural styles or patterns, and architectural design decisions and their rationale. This type of information is called Architectural Knowledge (AK) [13]. We assert that AK of OSS products can provide valuable insights to integrators, not only during the evaluation and selection phase, but also in later phases of the software lifecycle. However, there is no empirical evidence about the AK needs of OSS integrators. This motivated us to carry out an empirical study to investigate this topic. The contributions of this paper are an empirical understanding of:

- The architectural knowledge needs of OSS integrators;
- Why architectural knowledge is needed and how it can help OSS integrators;
- The availability of architectural knowledge to OSS integrators;
- The relative importance of architectural knowledge.

Based on these results, we also identify a number of opportunities for further research to bridge the gap between the OSS and SA research communities.

This paper proceeds as follows. Section 2 presents background and motivation of this study. Section 3 presents the design of this study. Section 4 presents results. Section 5 presents discussion and conclusion of this paper.

2 Background and Motivation

In this section, we present relevant related work and motivate our study. Section 2.1 provides an overview of CBSD with OSS. Section 2.2 introduces the field of SA and AK. Section 2.3 provides an overview of the research on SA in the context of OSS.

2.1 Component-Based Development with Open Source Software

OSS products have become more commercially viable as an alternative to COTS products [14]. In the last decade or so, OSS has been increasingly used by software-development organizations in CBSD [1]. While OSS components are often used as if they were closed-source components [11], there are some differences from an integrator's point of view. The availability of the source code eases component integration and allows white-box testing [15]. Di Giacomo reviewed the activities of CBSD and concludes that these activities need to be extended to accommodate the different nature of OSS [16]. Furthermore, the relationship with the supplier (COTS

vendor versus OSS community) is different; one reason is the fact that the OSS integrator does not pay an OSS community [17].

Li et al. list three main phases in the CBSD lifecycle: *selection*, *integration*, and *maintenance* [11]. The remainder of this subsection briefly reviews the literature on OSS-based CBSD according to these lifecycle phases.

Component selection. A critical success factor in CBSD is the selection of appropriate OSS products, and both researchers and industry have proposed a variety of OSS evaluation and selection methods [18]. However, studies have also shown that practitioners typically do not use these "normative" selection methods [19, 20]. Hauge et al. observed a *first fit* rather than a *best fit* strategy [19].

Component integration. The source code's availability allows direct modification by the OSS integrator [16], or "glue code" can be written to make integration easier. However, integrators typically do not make changes to the source code, for various reasons [11]. One obstacle is that it often requires an in-depth knowledge of the component to be able to make useful changes [11].

Component maintenance. Once OSS components are integrated, they must be maintained as part of an application. If custom changes were made during the integration phase, these may have to be separately maintained if the changes are not incorporated back into the source base that is maintained by the OSS community [21]. An OSS integrator can choose to actively participate in the development or roadmap planning of the OSS product [17, 21].

2.2 Software Architecture and Architectural Knowledge

The field of SA has emerged as an important sub-discipline within the research area of software engineering [22]. SA as a product has been recognized as an important design artifact in software development activities, including analysis, design and evaluation activities as well as implementation and evolution [23]. A system's SA typically constitutes the design decisions about a system [24], such as the use of certain architectural styles or patterns [22], e.g., layers or model-view-controller (MVC). Architecture patterns are common solutions to recurring system design problems [25], and affect the system-wide quality attributes (QAs, sometimes referred to as a system's *"ilities"*) such as performance and reliability [26]. For instance, a "layered" architecture is likely to improve maintainability, as it facilitates a clear separation of concerns. However, passing large numbers of "messages" (e.g., function calls) up and down a layer "stack" may negatively impact performance. Various patterns have been documented in detail, for instance, in [25].

Information such as used patterns, design decisions, etc. is referred to as AK. We note that there is no commonly accepted definition of AK. In recent years, the SA

research community has recognized the value of AK and has started to focus on recording, managing and sharing AK[1] [13].

2.3 Software Architecture in Open Source Software Research

Software architecture has received little attention in OSS research. The remainder of this section presents a brief overview of research on SA in the context of OSS.

Tran et al. [27] presented an experience report on "architectural drift", when concrete architecture (implementation) differs from the conceptual architecture (design). They argue that a system can be more easily understood by developers if its architecture is repaired and demonstrate their approach to architectural repair for two OSS products. Nakagawa et al. presented a case study of an OSS web-based system, and also found that the architecture had drifted after approximately two years of development [28]. They highlight that the architecture affects the product quality, and propose architecture refactoring to repair the architecture.

Capiluppi and Knowles report on architecture recovery of three OSS products in the same domain (instant messaging) [29] and found that a common architecture for these products had emerged. They argue that architecture recovery may facilitate OSS developers to understand the design as well as a sharing of tacit knowledge of other OSS developers.

Merilinna and Matinlassi investigated the state of the art and practice of OSS integration [20]; while they did not specifically focus on SA, they found that architectural mismatch was either preempted by selection of OSS components from a list of "fluently integrating" components, or that practitioners were not aware of the concept of architectural mismatch.

Matinlassi performed a survey to investigate the role of SA as perceived by 15 OSS communities, and found that "architecture" was mostly considered to be a high-level, coarse grained abstraction of the system's structure [30]. Modularity was considered to be the most important QA of an SA, whereas other QAs such as performance and ease of integration were not considered important by most respondents.

Ali Babar et al. [31] argued that organizations may be able to more confidently use OSS components in a software product line if OSS communities pay more attention to the architectural aspects during development and evolution of those components. Previously, researchers and practitioners have organized SA-related workshops [32] to draw attention to this topic in OSS, and proposed research roadmaps [33, 34]. Recently, AK Management (AKM) in OSS projects is discussed from the perspective of OSS developers [35].

So far, there have only been a few empirical studies on the role and importance of SA in OSS. These mostly focused on architectural repair and recovery. However,

[1] We are aware of the semantic difference between "knowledge" and "information". Even though "information" may be the appropriate term, we use the term "knowledge" as is custom in the software architecture community.

there has been no study that has focused on OSS integration from the OSS integrator's point of view, and in particular with respect to a component's AK.

2.4 Research Objectives

Hauge et al. identified a lack of empirical research in the OSS research area and highlighted the need for studies related to integration of OSS components [1]. The objective of our study was to empirically explore the observations and experiences of OSS integrators for gaining and disseminating insights into the architectural aspects of OSS integration.

Much of the literature on OSS selection highlights the importance of evaluation criteria such as the OSS license, support, functionality, and so on. However, there are few studies from the perspective of OSS integrators. Since software architecture plays an important role in component integration, it is important to investigate what architectural knowledge OSS integrators need. Therefore, research question one is:

RQ1: What AK of OSS products do integrators need?

Besides an understanding of *what* AK is needed, it is also useful to understand how AK can help OSS integrators. Hence, our second research question is:

RQ2: Why do integrators need AK of OSS products?

Furthermore, it is also important to find out whether the required AK about OSS component is usually available. Hence, our third research question is:

RQ3: Is AK of OSS products generally available to integrators?

In this paper, we assert that AK about an OSS product is important. However, it is equally important to find out OSS integrators' perceived importance of AK in integration. In order to get an empirically-based understanding of the importance of the role of architecture in OSS integration, we defined the fourth research question:

RQ4: What is the relative importance of AK of OSS components?

3 Research Design

3.1 Research Method and Data Collection

Since the role and importance of software architecture in OSS is still a largely unexplored research area in its nascent phase, a qualitative research approach is appropriate [36]. We decided to conduct an in-depth exploratory, qualitative survey using semi-structured interviews to collect data from practitioners. The use of

interviews gives a researcher the flexibility to go deeper into unforeseen types of information that may emerge during an interview [37].

We invited IT practitioners through our professional network to participate in our study; our sample was therefore a *convenience sample* [38]. Table 1 lists background information of the participants. All but one participant worked at organizations located in different countries in Europe; P5 was located in the USA. We do not report on specific details for confidentiality reasons. Two participants (P3 and P4) worked both at organization C, but at different branches or departments. We indicated this by a number suffix (i.e., C1, C2). The participants at organization E were contacted through our professional contact at this organization, who was also one of the participants. The participants had various positions, and all had extensive experience in the field. In total, we drew data from 12 participants who worked at five different organizations. Though participants P6-P12 all worked at organization E, they worked in different teams or departments (E1 to E6).

Table 1. Participants of our study.

ID	Position	Experience	Org.	Domain	Interview
P1	Software architect	13 years	A	Business	Phone
P2	Software developer	5 years	B	Embedded	Face-to-face
P3	R&D developer	13 years	C1	Telecom	Phone
P4	R&D developer	13 years	C2		Face-to-face
P5	Independent consultant	10 years	D	Business	Instant messenger
P6	Software architect	20 years	E1		Face-to-face
P7	Project leader	26 years	E2		Face-to-face
P8	Sr software designer	10 years	E3	Safety	Face-to-face
P9	Team leader*	25 years	E4	critical	Face-to-face
P10	Software architect	12 years	E5	systems	Face-to-face
P11	Technology manager	25 years	E2		Face-to-face
P12	Team leader*	15 years	E6		Face-to-face

* Participants have also experience as a software architect.

The participants worked at organizations in different domains that integrate OSS products in various degrees. Organization A is a large consultancy organization that develops business process support systems. Organization B develops mostly embedded software. Organization C is a research and innovation institution that develops both proof-of-concept prototypes and final products for customers. Participant P5 (organization D) is an independent consultant, and works on business process support systems. Organization E develops hardware and software for safety critical systems.

Prior to conducting the interviews, we designed an interview guide [39]. All face-to-face interviews were conducted at the premises of the participants' organizations. All but one interview lasted approximately one hour; the interview with P5 (through instant messenger (IM)) lasted approximately two hours. We digitally recorded all face-to-face and phone interviews with the participants' consent. All interviews were transcribed verbatim by the interviewer (the transcript

of the interview with P5 was recorded by the IM client), resulting in more than 120 pages text (A4 size).

3.2 Data Analysis

Due to the qualitative nature of the collected data, we chose to analyze the data using qualitative data analysis methods [37]. We thoroughly read all interview transcripts, during which we extracted phrases of interest that were relevant to one of our research questions. The extracted data was stored in a spreadsheet, along with the page number of the source transcript that allowed us to trace back phrases to their original context. We annotated each entry with a code that reflected the contents of the entry. After this, the data were sorted and grouped based on the codes. Per group, we used the constant comparison technique [37] to identify common themes for answering our research questions.

4 Results

4.1 RQ1: Architectural Knowledge Needs of OSS Integrators

We identified four categories of information that integrators would like to have. Table 2 presents an overview of the findings using the categories that emerged from the coding of the participants' answers. It is interesting to note that the first type of AK (N1) affects the other three types of AK (N2-N4); for instance, a component's structure directly affects its quality attributes and behavior. We address this in more detail in Section 5.

Table 2. Types of architectural knowledge needed by OSS integrators.

ID	Description	Reported by
N1	**Component structure.** Patterns, partitioning, structure	P1, P2, P3, P4, P5, P6, P7, P10, P12
N2	**Quality attributes and behavior.** Information about performance, e.g., bottlenecks, processor usage, disk usage, and other resources; reliability, stability, robustness, predictability, use of tactics.	P2, P4, P5, P6, P7, P8, P10, P11
N3	**Architectural fit.** Ease of integration, interface, API compliance, dependencies	P2, P3, P4, P5, P6, P7, P8, P10, P11
N4	**Component usage.** Insight in how the component can be used and how it performs the task	P2, P4, P10

4.1.1 Component structure

Most participants were interested in the internal structure of components. When speaking of a component's internal structure, participants often spoke of 'pattern'. In particular, some participants mentioned the presence of patterns such as layers and model-view-controller to be interesting for their purpose. One participant stated:

"Well, the first thing that I would do is to look at the documentation, in which they explain the structure there, preferably in pictures. And when those are not available, then it's a matter of extracting a zip file [jar file] and see how the directory structure would look like." (P4).

4.1.2 Quality attributes and behavior

Several participants explicitly indicated specific runtime QAs, such as performance, reliability and stability. Like the literature on QAs, the participants also agreed that these QAs are directly affected by a component's behavior related to the use of resources (e.g., memory, processor, interrupts, database connections). Integrators consider it to be important to understand how a component behaves regarding these system resources; one participant reported to have used a "sandbox" to measure performance, processor usage, disk usage, and similar parameters.

One factor that affects a component's behavior is the use of architectural tactics [26, 40]. A tactic is a common technique to achieve a certain quality attribute. For instance, in order to improve the performance of software that connects to a database system, a developer may apply the "connection pooling" tactic. Such information is valuable to integrators, as one participant described: *"if it is a component that uses relatively expensive or limited system resources, then you'd like to know the strategy of those components to deal with resources. If I'm integrating it and it's using 10 out of 11 available database connections […] that's certainly handy to know."* (P10).

We did not quantitatively analyze the most important quality attributes such as in [41], as our sample size was limited, and QAs are likely to be dependent on the domain in which software operates.

4.1.3 Architectural fit

The most common concern of participants was the architectural fit of a component; in other words, does a component fit in the existing architecture? One participant described it clearly: *"[I look at] whether it can be used in the architecture that I had envisioned. To what extent do I need to adapt my own software in order to be able to use that product? [And] if I have to make certain changes in an OSS product, in order to be able to use it, how much."* (P3).

Participants used the term "architectural fit", but also referred to the interface and API of a component. It was interesting to find that a large majority of the participants distinguished a component's architectural fit from its internal structure or used patterns (see subsection 4.1.1), and generally considered the former to be more important than the latter. This confirms common knowledge that patterns have a direct influence on a component's architectural fit [42].

4.1.4 How to use a component

A few participants mentioned the need to understand how they can use a component. Besides insight in whether the component fits (see subsection 4.1.3), practitioners need to understand how it can be used within the system they build, and how to access the functionality provided. The participants prefer to have examples of how to use the software, or even the availability of a test environment that demonstrates how

a component can be used, as one participant suggested: *"If the component came together with a sort of test environment, a sort of test application, that could show how the components can be used, and how it performs the tasks that it is supposed to do. That would help a lot."* (P2).

4.2 RQ2: Why is Architectural Knowledge Needed?

The previous section addressed the question *what* AK integrators need; in this section we addresses the question *why* integrators need AK. We identified a number of different reasons how such knowledge can help integrators. Table 3 presents an overview of the findings to answer this question.

Table 3. Reasons why architecture knowledge is important to integrators.

ID	Description	Reported by
W1	**Quality assessment.** Architecture, partitioning and patterns indicate certain maturity and qualities and help to analyze a component from a performance and functional perspective.	P2, P5, P7, P8, P10
W2	**Assess architectural fit.** Internal structure and patterns indicate what architecture is used and whether it fits in the existing architecture	P1, P2, P3, P6, P7, P10
W3	**Improve maintainability.** Patterns improve maintainability and replacing of components	P1, P2, P7, P8, P10, P12
W4	**Help to use component.** Architectural information and patterns help you to use the component in a useful way	P1, P10

4.2.1 Quality assessment

An important use of AK is that it can be used in the assessment of a component with respect to its quality attributes. Participants particularly referred to the structure and patterns used in this context. Patterns are common solutions with predictable effects on a component's QAs. This could refer to a product's runtime attributes, such as performance and reliability, but also with respect to its build-time properties, such as the ease of integration. One participant explained: *"[Information of internal structure is] certainly very useful information, because such a pattern indicates what kind of architecture was used, and such a pattern indicates a certain quality. That could be configurability, but also the ease of integrating, those patterns secure a certain stability."* (P7).

4.2.2 Architectural fit assessment

Several participants indicated that architectural information provides useful insights into whether or not an OSS product is compatible at the architectural level with the main system in which it may be integrated. As noted by one participant: *"Well at least it indicates what style is being used, and what are the fundamental concepts, and your own product, or platform in our case, also has certain styles, and then you can see, whether it fits together. Does it make sense to try to put it together, or will we need a whole lot of glue code to be able to use it. And if so, is it worth our while,*

or is it easier to develop ourselves. At a high level it indicates whether something is a good match with your product, with the existing software." (P10).

4.2.3 Maintenance and evolution

A number of participants indicated that knowledge of the patterns used in a component could help in the maintenance phase of a product, including defect fixing. Patterns increase the understandability of how a component is constructed, which makes it easier to change the component. One participant described this as follows: *"If you have some OSS that's constructed using good patterns, then of course it'll help you to make changes. If you get OSS that's constructed in an ad-hoc manner, organically grown, and you decide to make changes to make it fit better, then that's completely useless. The more it is structured, thought about it in terms of layers and standard structures, the easier it will be to adopt it and change it."* (P12).

While the actual presence of patterns improves a product's maintainability, it is important to understand *which* patterns have been used, in order to make changes that do not degrade the pattern's integrity and consistency.

4.2.4 Using the component

A good knowledge of architectural aspects of a component can help an integrator to understand the appropriate use of that component. During our study, a few participants also indicated that AK such as the patterns used provides valuable insights into whether or not a component will be suitable in the context of a given system and how to use that component. One of the participants explained: *"if you wouldn't know those [architectural concepts and patterns], it's difficult to use such a framework in a useful way. So in that case it would be difficult to find everything in the code, so at least you'd like to see some high level descriptions of the concepts and design patterns."* (P10).

4.3 RQ3: The Availability of Architectural Knowledge

In this section, we address the question whether or not AK is usually available to OSS integrators. We identified two themes, namely (1) the general availability of AK and (2) how integrators deal with the lack of AK. We decided to present this analysis in a descriptive way, rather than a tabular representation as used in 4.1 and 4.2.

4.3.1 Availability differs per product

The results indicate that whether or not AK is available, depends on the product. One participant highlighted that: *"[for] open source it's quite easy to figure out how it fits into my architecture. If I look at Spring, how does this fit into my architecture, is it MVC or part of MVC… I also think that is indirectly a big plus for OSS, because they actually need to be compatible, they need to be able to integrate with other parts at the architecture level, otherwise they wouldn't survive."* (P1). This suggests that AK is sufficiently available. Another participant confirmed that for well-known

frameworks such as the Spring framework[2], this information is indeed quite easily available, however, this may not be the case with other components: *"in case of Spring yes, but in other components not so much, and you just have to look in the documentation and in some cases even in the implementation."* (P3).

Participants P5 and P7 explicitly claimed that there is typically not much AK of OSS products available. One participant emphasized that a good design will result in good quality code, and makes the component author's intents clear: *"I don't usually have much architectural info when investigating an OSS component, unless it's in the docs, on the website etc. Historically, there hasn't been a focus on architecture patterns by OSS component authors. It's not a best practice to include architectural patterns information."* (P5)

4.3.2 How integrators deal with a lack of architectural knowledge

The participants of our study mentioned different approaches to deal with a lack of AK of an OSS product. We present the main findings under the following categories:

Look at others. One way to assess the quality of a product's QAs, is to look at other customers of the component, as explained by one participant: *"Our line of thought with respect to the 'ilities' was, as long as a rather large group of people uses it, you may assume that most of the trouble with the 'ilities' are solved."* (P6).

Assume the worst. Another approach to deal with a component's characteristics, such as security, is to assume the worst, and make sure that the rest of the system compensates for its shortcomings: *"I don't think I would try to understand whether a certain OSS product, whether it's secure enough. I think I would assume that it's not secure enough and then I would make sure that the environment in which the software is run takes care of the security."* (P3).

Try to recover. If architectural information in the documentation is missing, then the only solution to this is to look into the implementation or to ask the community for more information. However, in such a case, the time to get a reply was considered to be very important due to development schedules and deadlines. Furthermore, while studying the implementation was mentioned as a solution, it should be considered as a last resort: *"When you have to go into the code to know how it works, then I do think you have a problem. […] I think open source has much potential but the investment in knowledge is quite an issue."* (P9).

4.4 RQ4: The Relative Importance of Architectural Knowledge

Our last research question investigates the participants' perceived importance of AK. This puts the need for AK in perspective compared to other selection criteria, which are known to be important, such as availability of support and license. One participant considered AK as just one type of information that is needed, but stated that: *"the more you know, the better"* (P10). As for RQ3, we identified a few themes, which we discuss next.

[2]http://www.springsource.org/

4.4.1 Missing AK hinders usage

In subsection 4.3.1, participant P1 indicated that the availability of AK of an OSS product is important for the project's survival. Three participants explicitly indicated that the lack of an OSS component's AK means that it drops on the list of candidates, and hinders its usage. This means that a lack of AK negatively affects the selection decision of whether to use the OS product. One participant explained: *"Well, then you're making a big investment if you do that. So I can think it can hinder you in using it. When you have to go into the code to know how it works, then I do think you have a problem."* (P9).

4.4.2 Need for AK depends on product type and size

A recurring factor that influenced the participants' interest in having AK of an OSS product is the *type* and *size* of the component. The information needs heavily depend on whether it is a library that provides functionality or non-critical parts, such as graphical widgets, or that it is a central component that makes up an important part of a system. One participant described it as follows: *"If I'm looking for an MVC framework, then yes [I'll be interested in architectural information]. But if I'm looking for a foo-munging module, and there are 12, then the last 3 release dates, smoke reports, browsing source, etc., are likely more important.*" (P5).

A few participants explicitly highlighted the architectural impact that an OSS component may have on the existing system architecture. One participant described: *"But for instance the reporting engine, if we choose product A or B, that was more of a feature level, and not typically on the architecture level, even if we would pick product A or B, it wouldn't ruin our existing architecture because this is more on the side."* (P1).

Related to this issue is whether practitioners value a component's *functionality* or *architecture*. On the one hand, some participants highlight the focus on functional compliance; one participant explained this concisely: *"My software MUST meet the functional requirements, and MAY have a good architecture."* (P5, emphasis by participant in IM transcript).

Furthermore, practitioners' interest in a component's functionality or architecture also depends on the type of the component. One participant explained: *"Well, frankly it's always been the functionality for us than that we're looking at the architecture. That may have to do with the type of components that we integrate. They're typically not complete subsystems, but rather limited libraries. Just those things that don't really count."* (P10).

On the other hand, some participants preferred a well-designed component with a good architecture to the functionality provided by the component. Extensibility and size of the component are decisive factors, as one participant explained: *"I would tend to choose for good quality and architecture, but perhaps a lack of functionality. But there should be a good possibility to extend that functionality, and when that's not there, then I may decide to take a chance and select the thing with all functionality but less quality, and to fix that thing myself. That also depends on the size by the way, when it's a huge project, then I won't start on that."* (P4).

4.4.3 The relative importance of architectural knowledge

So far, the answer to the question whether or not AK is an important factor is: "it depends". Factors are the type and size of a component (e.g., library versus framework). In general, the participants of our study seem to be quite interested in a component's AK, which can provide valuable insights that may affect the decision to use the component. One participant summarized this as follows: *"It depends on what it is, but I think if it is something that you're interested in anyway, in how it's constructed, which can be part of the evaluation of the piece of open source, then it's certainly interesting. It can give you a good feeling that people have thought about it. It fits or doesn't fit with what we have. So yes, it's certainly valuable."* (P10).

The participants generally agreed that the quality of a component can have many aspects, but that the critical selection factors depend on the context. One participant highlighted the importance of the implementation language in these words: *"The use of patterns in OSS components played some part when considering their use, but the flexibility of Perl allows integration of widely-varying programming models, so for Perl components, the test coverage, documentation and community support were more important."* (P5).

5 Discussion and Conclusion

In Section 4, we have identified the types of AK of OSS needed by OSS integrators and the main reasons for why the AK is important. It is interesting to note that the categories identified in Section 4.1 (*what* AK is needed) and 4.2 (*why* is AK needed) are quite similar. Participants indicated the need of understanding a component's structure, its quality attributes and behavior regarding e.g., system resources, whether or not components are an architectural "fit" with the main system's design, and how to use a component. Following are the main reasons *why* OSS integrators consider architectural knowledge of a component to be valuable:

- Assess the quality of the component;
- Assess the architectural fit of the component;
- Affect the maintainability of the component;
- Understand how the component can be used.

Knowledge of a component's structure (including its patterns) seems to be the most important aspect for integrators. While the participants indicated a desire to know a component's QAs, its architectural fit (compatibility), and how to use it, it seems that the architectural structure of a component is valuable input to satisfy most AK needs. A component's architecture structure, including its patterns, directly affects its QAs, architectural compatibility, and can provide insight on how to use the component.

The four categories of why an integrator would like to have AK of a component can be mapped to the three main phases in CBSD briefly outlined in Section 2.1. Quality assessment and assessment for architectural fit are both activities that are performed in the *Evaluation* phase, when components are evaluated and selected. Understanding of how to use a component is important in the *Integration* phase, when components are integrated into a product. Finally, having AK to improve a

component's maintainability supports the *Maintenance* phase of OTS integration, after the main system has been deployed. Therefore, we can conclude that AK can support the OSS integrator in all three phases of CBSD.

However, in Section 4.3 we found that availability of AK of OSS products may vary, and just how much AK is needed also varies. When AK is not available, OSS integrators typically do not try to recover it, which means they have to take a different strategy to deal with such lack of information. From Section 4.4 it has become clear that practitioners do consider AK to be valuable, but that depends on the type and size of the product. A lack of AK, however, was shown to be a potential obstacle for using a component. These results highlight the importance to investigate how OSS integrators can be supported. We suggest a research agenda along the following lines:

- We found that the type and size of a product affect whether or not an OSS integrator needs to acquire AK of the product. It would be valuable to gain a deeper insight into how these characteristics affect the need for AK, what other factors are at play, and to develop a systematic method to assist OSS integrators in identifying what AK is needed and how AK can be identified in an effective way.
- Some OSS projects are more successful in publishing AK of the product than others, and one of our participants suggested that it is vital for a project's survival. What factors affect whether a community makes such AK available, and how can other OSS communities be supported in this activity?
- Our results suggest that if AK is not available, OSS integrators do not try hard to recover it. Furthermore, looking into the source code seemed to be the only solution. However, it would be valuable to, based on a larger scale survey, get better insights into whether OSS integrators recover AK, how they use this AK, whether they store it, and whether this AK is contributed back to the OSS community.
- Related to using OSS components is the integration of so-called "Inner Source Software" (ISS); ISS is closed-source software developed within an organization that has adopted OSS development methods, a phenomenon known as "Inner Source" [43]. In Inner Source, departments can be consumers and producers in an internal software market ("bazaar"). It would be very informative to understand what kind of AK integrators need and have available in Inner Source, and what lessons can be drawn from Inner Source to use in OSS integration (and vice versa).

5.1 Limitations of this Study

We are aware of a few limitations of this study. Firstly, we based this paper on data gathered through 12 interviews, which is insufficient to draw general conclusions. However, since the role of SA has not been studied extensively in the context of OSS integration, we decided to perform an *exploratory* study. Once this field has matured and specific hypotheses have been defined based on initial findings, we argue that

other types of studies with larger numbers of participants will be more appropriate, such as questionnaire-surveys.

Secondly, our sample of participants was a convenience sample, which means there is a selection bias. Participants were contacted through our professional network. Furthermore, seven participants worked at one organization that is active in a safety critical domain; this may have biased the participants' views towards certain concerns. However, we did not find significant differences with respect to the participants' opinions and needs of AK. We argue that, since this is an exploratory study, these results can be used as input to identify hypotheses to design studies based on larger samples of participants from a variety of product domains.

5.2 Conclusion

This paper presents the results of an exploratory interview-based survey of software architects and other IT professionals to investigate the importance of architectural knowledge (AK) in the integration of OSS products CBSD. In particular, this paper presents a classification of different types of AK that is considered to be useful, why AK is needed, whether AK is available, and the relative importance of AK in the integration of OSS components. Knowledge of a component's partitioning and its patterns used seems to be particularly important to satisfy the various reasons of why AK is needed (e.g., assessment of quality, architectural fit, etc.).

Despite an increasing attention for SA in software engineering, SA has received little attention in the OSS research community. This paper provides empirical findings to demonstrate that integrators have a need for AK of OSS products. Based on our findings, we suggested a number of open research questions in order to bridge the gap between the OSS and SA research communities.

Acknowledgements. This work is partially funded by IRCSET under grant no. RS/2008/134 and by Science Foundation Ireland grant 03/CE2/I303_1 to Lero (www.lero.ie). We are grateful to the participants of our study for their time and enthusiasm.

6 References

[1] Ø. Hauge, C. Ayala, R. Conradi, Adoption of Open Source Software in Software-Intensive Organizations - A Systematic Literature Review, Information and Software Technology, 52 (11), 2010, pp. 1133-1154.
[2] Ø. Hauge, C.-F. Sørensen, A. Røsdal, Surveying Industrial Roles in Open Source Software Development, in: J. Feller, B. Fitzgerald, W. Scacchi, A. Sillitti (Eds.) Open Source Development, Adoption and Innovation, Springer, 2007, pp. 259-264.
[3] D. Garlan, R. Allen, J. Ockerbloom, Architectural mismatch: why reuse is so hard, IEEE software, 12 (6), 1995, pp. 17-26.

[4] S.A. Hissam, C.B. Weinstock, Open Source Software: The Other Commercial Software, in: J. Feller, B. Fitzgerald, A. van der Hoek (Eds.) 1st Workshop on Open Source Software Engineering, collocated with the 23rd International Conference on Software Engineering (ICSE), 2001.
[5] L. Morgan, P. Finnegan, Benefits and Drawbacks of Open Source Software: An Exploratory Study of Secondary Software Firms, in: J. Feller, B. Fitzgerald, W. Scacchi, A. Sillitti (Eds.) Open Source Development, Adoption and Innovation, Springer, 2007, pp. 307-312.
[6] C. Ayala, Ø. Hauge, R. Conradi, X. Franch, J. Li, K.S. Velle, Challenges of the Open Source Component Marketplace in the Industry, in: C. Boldyreff, K. Crowston, B. Lundell, A.I. Wasserman (Eds.) Open Source Ecosystems: Diverse Communities Interacting, Springer, 2009, pp. 265-271.
[7] W. Chen, J. Li, J. Ma, R. Conradi, J. Ji, C. Liu, An empirical study on software development with open source components in the chinese software industry, Software Process: Improvement and Practice, 13 (1), 2008, pp. 89-100.
[8] A. Jaaksi, Experiences on Product Development with Open Source Software, in: J. Feller, B. Fitzgerald, W. Scacchi, A. Sillitti (Eds.) Open Source Development, Adoption and Innovation, Springer, 2007, pp. 85-96.
[9] T.R. Madanmohan, R. De', Open source reuse in commercial firms, IEEE Software, 21 (6), 2004, pp. 62-69.
[10] C. Ayala, Ø. Hauge, R. Conradi, X. Franch, J. Li, Selection of third party software in Off-The-Shelf-based software development: An interview study with industrial practitioners, The Journal of Systems and Software, 84 (4), 2011, pp. 620-637.
[11] J. Li, R. Conradi, O.P.N. Slyngstad, C. Bunse, M. Torchiano, M. Morisio, Development with Off-the-Shelf Components: 10 Facts, IEEE software, 26 (2), 2009, pp. 80-87.
[12] J. Bosch, J.A. Stafford, Architecting Component-Based Systems, in: I. Crnkovic, M. Larsson (Eds.) Building Reliable Component-Based Software Systems, Artech House Publishers, Norwood, MA, USA, 2002.
[13] M. Ali Babar, T. Dingsøyr, P. Lago, H. van Vliet, Software Architecture Knowledge Management: Theory and Practice, Springer, 2009.
[14] B. Fitzgerald, The transformation of open source software, MIS Quarterly, 30 (3), 2006, pp. 587-598.
[15] P. Mäki-Asiala, M. Matinlassi, Quality Assurance of Open Source Components: Integrator Point of View, in: Proceedings of the 30th Annual International Computer Software and Applications Conference (COMPSAC), 2006, pp. 189-194.
[16] P. Di Giacomo, COTS and Open Source Software Components: Are They Really Different on the Battlefield?, in: X. Franch, D. Port (Eds.) COTS-Based Software Systems, LNCS 3412, Springer Berlin / Heidelberg, 2005, pp. 301-310.
[17] J.S. Norris, Mission-critical development with open source software: Lessons learned, IEEE software, 2004.
[18] K. Stol, M. Ali Babar, A Comparison Framework for Open Source Software Evaluation Methods, in: P. Ågerfalk, C. Boldyreff, J.M. González-Barahona, G.

Madey, J. Noll (Eds.) Open Source Software: New Horizons, Springer, 2010, pp. 389-394.
[19] Ø. Hauge, T. Østerlie, C.-F. Sørensen, M. Gerea, An Empirical Study on Selection of Open Source Software - Preliminary Results, in: A. Capiluppi, G. Robles (Eds.) 2nd workshop on Emerging Trends in FLOSS Research and Development, collocated with the International Conference on Software Engineering (ICSE), Vancouver, Canada, 2009.
[20] J. Merilinna, M. Matinlassi, State of the Art and Practice of Open Source Component Integration, in: Proceedings of the 32nd Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 2006, pp. 170-177.
[21] K. Ven, H. Mannaert, Challenges and strategies in the use of Open Source Software by Independent Software Vendors, Information and Software Technology, 50 (9-10), 2008, pp. 991-1002.
[22] M. Shaw, D. Garlan, Software Architecture: Perspectives on an Emerging Discpline, Prentice-Hall Inc., New Jersey, 1996.
[23] A. Tang, P. Avgeriou, A. Jansen, R. Capilla, M. Ali Babar, A comparative study of architecture knowledge management tools, Journal of Systems and Software, 83, 2010, pp. 352-370.
[24] A. Jansen, J. Bosch, Software Architecture as a Set of Architectural Design Decisions, in: 5th Working IEEE/IFIP Conference on Software Architecture (WICSA), Pittsburgh, PA, USA, 2005, pp. 109-120.
[25] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal, Pattern-oriented Software Architecture - A System of Patterns, J. Wiley and Sons Ltd., 1996.
[26] L. Bass, P. Clements, R. Kazman, Software Architecture in Practice, 2nd ed., Addison-Wesley, Boston, MA, USA, 2003.
[27] J.B. Tran, M.W. Godfrey, E.H.S. Lee, R.C. Holt, Architectural repair of open source software, in: Proceedings of the 8th International Workshop on Program Comprehension (IWPC), 2000.
[28] E. Nakagawa, E. de Sousa, K. de Brito Murata, G. de Faria Andery, L. Morelli, J. Maldonado, Software Architecture Relevance in Open Source Software Evolution: A Case Study, in: Proceedings of the 32nd International Computer Software and Applications Conference (COMPSAC), 2008, pp. 1234-1239.
[29] A. Capiluppi, T. Knowles, Software Engineering in Practice: Design and Architectures of FLOSS Systems, in: C. Boldyreff, K. Crowston, B. Lundell, A.I. Wasserman (Eds.) Open Source Ecosystems: Diverse Communities Interacting, 2009, pp. 34-46.
[30] M. Matinlassi, Role of Software Architecture in Open Source Communities, in: Proceedings of the Sixth Working IEEE/IFIP Conference on Software Architecture (WICSA), Mumbai, India, 2007.
[31] M. Ali Babar, B. Fitzgerald, P.J. Ågerfalk, B. Lundell, On the Importance of Sound Architectural Practices in the Use of OSS in Software Product Lines, in: Second International Workshop on Open Source Software and Product Lines, collocated with the 11th International Software Product Line Conference, 2007.

[32] M. Ali Babar, B. Lundell, F. van der Linden, A Joint Workshop of QACOS and OSSPL, in: C. Boldyreff, K. Crowston, B. Lundell, A.I. Wasserman (Eds.) Open Source Ecosystems: Diverse Communities Interacting, Springer, 2009, pp. 357-358.
[33] C. Lennerholt, B. Lings, B. Lundell, Architectural issues in Opening up the advantages of Open Source in product development companies, in: Proceedings of the 32nd Annual IEEE International Computer Software and Applications Conference, IEEE Computer Society Washington, DC, USA, 2008, pp. 1226-1227.
[34] B. Arief, C. Gacek, T. Lawrie, Software architectures and open source software-where can research leverage the most?, in: J. Feller, B. Fitzgerald, A. van der Hoek (Eds.) 1st Workshop on Open Source Software Engineering, collocated with the 23rd International Conference on Software Engineering (ICSE), 2001.
[35] I. Stamelos, G. Kakarontzas, AKM in Open Source Communities, in: M. Ali Babar, T. Dingsøyr, P. Lago, H. van Vliet (Eds.) Software Architecture Knowledge Management: Theory and Practice, Springer, 2010, pp. 199-215.
[36] A.C. Edmondson, S.E. McManus, Methodological Fit in Management Field Research, Academy of Management Review, 32 (4), 2007, pp. 1155-1179.
[37] C.B. Seaman, Qualitative methods in empirical studies of software engineering, IEEE Transactions on Software Engineering, 25 (4), 1999, pp. 557-572.
[38] C. Robson, Real World Research: A Resource for Social Scientists and Practitioner-Researchers, 2nd ed., Blackwell Publishing, 2002.
[39] S.J. Taylor, R. Bogdan, Introduction to Qualitative Research, John Wiley & Sons, New York, 1984.
[40] N.B. Harrison, P. Avgeriou, How do architecture patterns and tactics interact? A model and annotation, The Journal of Systems & Software, 83 (10), 2010, pp. 1735-1758.
[41] D. Ameller, X. Franch, How Do Software Architects Consider Non-Functional Requirements: A Survey, in: R. Wieringa, A. Persson (Eds.) Requirements Engineering: Foundation for Software Quality, Springer Berlin / Heidelberg, 2010, pp. 276-277.
[42] M. Shaw, Architectural issues in software reuse: it's not just the functionality, it's the packaging, SIGSOFT Softw. Eng. Notes, 20 (SI), 1995, pp. 3-6.
[43] J. Wesselius, The Bazaar inside the Cathedral: Business Models for Internal Markets, IEEE software, 25 (3), 2008, pp. 60-66.