Software Interoperability Analysis in Practice – A Survey

H. ABUKWAIK and D. ROMBACH, University of Kaiserslautern

Software interoperability property plays a vital role in enabling interoperation in today’s system-of-systems, cyber-physical systems, ecosystems, etc. Despite the critical role of interoperability analysis in enabling a successful and meaningful software interoperation, it is still facing challenges that impede performing it effectively and efficiently. We performed an online survey of software engineers with software integration experiences to identify the main difficulties of performing interoperability analysis. The results confirm that the state of available practical support and current input artifacts used during the analysis are significantly perceived as important difficulties. Respondents claim a lack of guidelines and best practices for applying interoperability analysis and claim insufficiency of shared information about interoperable software units. This indicates the need for providing directive and rigorous guidelines for practitioners to follow and to enrich the content of shared documents about interoperable software units.

CCS Concepts: • Software and its engineering → Interoperability; Empirical software validation
Additional Key Words and Phrases: Survey, Interoperability analysis

1 INTRODUCTION
In the past, a software system was developed by a single organization to provide a tightly focused support for certain tasks and specific purposes. However, today’s software providers are urged to adopt integration solutions of independent software systems built by different organizations [4] [2]. Successfully integrating software units and enabling their meaningful exchange of data and services requires a comprehensive analysis of both their technical constraints (which control the actual exchange of data and services like network protocol, programming languages, data types, etc.) and their conceptual constraints (which control the meaningfulness of interoperation results like usage contexts, architectural constraints, semantics, qualities, etc.). Thus, interoperability analysis supports an early detection for software mismatches and estimating their impact on the desired interoperation. It supports decision making about the feasibility of completing an integration project within available budget and time.

However, according to Krueger [12], organizations adopting integration report success limitations due to the non-technical issues. For example, COTS-based integration projects often suffer from significant overruns in budget and schedule due to unexpected interoperability mismatches [3]. This indicates problems in performing interoperability analysis. In-depth investigation of these problems is important in order to provide better support for practitioners to increase the quality of the interoperability analysis results.

This paper contributes the identification of problems experienced when analyzing the interoperability between two software units from the perspective of software engineering practitioners with practical experience in integration. We focused on difficulties related to the practices along with available methodological support and the input artifacts used in performing the interoperability analysis. To reach a large number of practitioners, we designed and performed an online survey at the mid of 2016. The sample consisted of 64 software engineer with experience in software integration.

In Section 2, we briefly introduce the concept of software interoperability and its related analysis activity and input/output artifacts. In Section 3, we describe the design of the online survey, while in Section 4, we present the results of the survey along with our discussion. Finally, In Section 5, we summarize the work.

Authors’ addresses: H. Abukwaik and D. Rombach, abukwaik | rombach@cs.uni-kl.de, University of Kaiserslautern, P.O. Box 3049, 67653 Kaiserslautern, GERMANY
2 INTEROPERABILITY BACKGROUND

Interoperability is a key property of software units to cope with current business demands and property is defined by IEEE [11] as "the ability of two or more systems or components to exchange information and to use the information that has been exchanged". Beyond the exchange of information, we emphasize the importance of having the integrated software units conceptually and architecturally aligned to allow meaningful and proper exchange of data and services. In practice, interoperability is considered as the enabler for software integration [14] and determines the readiness of a software unit for integration.

Having this been said, building successful integration and consequently achieving meaningful interoperation starts with an early assessment and reasoning about the properties of the two software units that are intended to interoperate. This assessment takes place before adapting, configuring, or glue-coding the units [5] and it is performed by software architects and analysts. We call it ‘interoperability analysis’ and we define it as the process of checking the constraints and assumptions of two software units in order to find if they have any mismatches that impede their desired communication or the meaningful interoperation between them. The input artifact for interoperability analysis of external software units in black-box context (in which source code is not available) is usually the shared documentation by the owner (e.g., API documentation of Web Services). This input is used to retain and communicate information about the various aspects of the unit to its audience [9]. Thus, a proper software documentation is considered as a necessity for enabling software integration as it helps in assessing the software unit with reasonable effort [15]. With regard to the output of the interoperability analysis, it is a list of constraints and assumptions of the analyzed software unit. Such an output is used to compare it with the constraints and assumptions of the software system (which the analyzed unit is desired to interoperate with) in order to find if they have any mismatches.

3 RESEARCH METHODOLOGY

3.1 Research Goal and Questions

It is important to collect the practitioners’ experience on the current state of interoperability analysis to get insights about its main difficulties. Hence, we decided to perform a systematic explorative study as an online survey. This type of studies is appropriate for approaching a large number of participants and allows collecting a wide range of needed data (e.g., actual experiences or personal opinions) [8] [10].

We formulate our survey goal in terms of GQM goal template [17] as follows:

**Goal**: to explore the state of practice of interoperability analysis for external software units for the purpose of characterizing its current state and identifying its difficulties with respect to its practices and input artifacts in the context of a survey from the viewpoint of architects and analysts as the basis for developing practically applicable enhancements towards efficient and effective interoperability analysis.

We translate this goal into the following research questions (RQs):

**RQ1**: How is interoperability analysis been currently performed by practitioners?

**RQ2**: What are the difficulties experienced when performing interoperability analysis?

**RQ2.1**: with respect to its current practices?

**RQ2.2**: with respect to its input (i.e., available sources of information about external software units)?

3.2 Survey Design and Execution

This survey is designed according to the proposed guidelines by [7], [8] [10], and [13].

**Target Group**. The target group of our survey consists of architects and software engineers with practical experience in interoperability analysis within software integration projects (this include projects that have integration as a part of it). Accordingly, we invited 115 practitioners known for their experience in software integration directly using emails and asked for distributing the call to whom it deemed appropriate. We also posted the call for experts’ participation on web pages of professional groups (e.g., LinkedIn special groups for architecture, software interoperability, and software integration).
Questionnaire Structure. Based on our goal and research questions, we designed a questionnaire with a supportive 2-minute video that introduces the interoperability analysis concepts and terminologies that we use to ensure correct mutual understanding. The questionnaire covers the following aspects:
- Current state of interoperability analysis practices and input artifacts (RQ1), i.e., the respondents’ practical feedback on performing the interoperability analysis task with regards to its engineering aspects.
- Perceived difficulties (RQ2 including RQ2.1 and RQ2.2), i.e., the main difficulties that practitioners experience when analyzing the interoperability of external software units especially on the conceptual level.
- Demographic information, i.e., respondents’ experience in performing interoperability analysis, type of applications they build and integrate, their position, organization type, sector, size, and location.

Type of questions. The questionnaire consists of fixed-set answer questions some of single- and some of multi-selection. To avoid closing off the range of potential answers that respondents may provide, some questions have a choice to enter a free text answer if none of the provided selections applies. Also, some questions have the “I don’t know” to avoid forcing answers with the lack of knowledge.

Questionnaire length. The designed questionnaire includes up to 26 questions and we used filters to avoid overwhelming the participants with irrelevant questions with their experiences. The questionnaire needs between 10 and 15 minutes to answer them. The final questionnaire is available in the survey web page.

Pre-execution evaluation. To assess the survey quality, two senior software engineers with expertise in interoperability and architecture peer-reviewed it to check the relevance and clarity of the questionnaire and its supporting video. After revising the survey according to the review results, one expert in empirical software engineering assessed the questions with respect to the principles defined in [13], and the ethical criteria stated in [1]. Accordingly, questions were re-categorized and shortened for understandability.

Implementation. We implemented the survey using the Limesurvey [16] and it was deactivated after six weeks. The final dataset is stored in a repository of the AGSE group of Kaiserslautern University. If you are interested in further anonymized analysis results, please contact the author.

Pilot study. We performed a pilot study with four software engineering researchers (with a background in interoperability) and two computer science researchers (with no background in interoperability) to assess the understandability of the questions and to estimate the required time to answer them. We encouraged the six researchers to take notes on any ambiguous words, uncertainties about the meaning of the questions or their offered answers, and to track the time they spend in filling the questionnaire. Accordingly, two questions were classified as very complex, so we split them into logical subgroups.

3.3 Data Analysis
We analyzed the data using MS Excel and IBM SPSS Statistics 23 [6]. Our descriptive analysis includes the median and frequency. As some of the questions were presented to respondents conditionally, we explicitly report the total number of subjects (N) who answered each question. We perform statistical analysis to explore how significantly different the ordinal answers are from a specific point using the One-Sample Wilcoxon signed-rank test. Also, we analyze the statistical difference between two groups of respondents based on their experiences using Pearson’s Chi-Square (χ²) test for binary data and Mann-Whitney (U) test for ordinal data. In addition, we run Spearman rho (ρ) test to check the correlation between ordinal variables.

4 RESULTS AND DISCUSSION
In total, we got 73 complete responses for our survey from the targeted group. However, we excluded nine responses as their demographic information showed they neither had any year of experience in integration nor they worked on any integration projects. This ensures credible findings based on actual experiences rather than inexperience opinions. Accordingly, the final included respondents size was N = 64.

---

2 https://www.youtube.com/watch?v=-WUU8sQmn08
3 http://wwwagse.informatik.uni-kl.de/staff/abukwaik/pub/EASE17/SurveySoP.htm
Password: ease17. After publication, it will be available under sharing agreement.
4.1 Background Overview

Software integration experience. The respondents’ experience in software integration varied in terms of experience years and the number of projects they participated in (see Fig. 1). The biggest shares (35.9 and 30.2) of years of experience were for “2-5 years” and “> 8 years” respectively. They participated mostly (25.4) in more than 5 projects and played different roles across the different projects like programmers (71.4), architects (54), system analysts (39.7), project managers (34.9), and testers (23.8).

Fig. 1 Respondents’ experience in software integration.

Nature of integration projects. The respondents worked mainly (80.8) in the domain of Information Systems, Mobile Systems (30.1), and Embedded Systems (26). They also reported experiences in integrating Open Source Software (OSS), Commercial-of-the-Shelf (COTS), Web Service APIs, and Platform APIs all with almost equal shares (54.5 on average for each type). The projects’ size was mainly (40.6) medium (i.e., 3 - 6 months, $250-750K, 4 - 10 team members) and the rest share was divided almost equally between large (i.e., > 6 months, > $750K, > 10 team members) and small (i.e., < 3 months, < $250K, 3 - 4 team members).

Nature of work organizations. While, the respondents were mostly (37.1) employed by enterprises (i.e., have > 1000 member), very few (only 8.1) were employed by large organizations (i.e., 251 - 999 members). The others were equally distributed (27.4) between small (i.e., have < 50 members) and medium (i.e., 51 - 250) organizations. Among the so many reported sectors (e.g., agriculture, health, finance, military, automotive, etc.), the software development sector was dominant. The data was collected internationally from several locations (e.g., USA, UK, Switzerland, Belgium, etc.), but the majority (32.8) was from Germany.

4.2 Interoperability Analysis As-Is State in Practice (RQ1)

Here, we characterize the current state of practice with regards to the performed interoperability analysis.

Neglected interoperability analysis. We surprisingly found that a large group (42.2) of the respondents stated that interoperability analysis took no place at all in their integration projects. The respondents (N = 27; 42.2) attributed this behavior to different reasons including:

- R1: There is no enough knowledge and experience about it %37,
- R2: Priority is given to other tasks, e.g., implementation and testing %37,
- R3: Tight schedule and limited resources %29.6, (R4) It is hard to perform %18.5,
- R5: Unit and integration testing are performed instead %18.5, and
- R6: It is not that necessary %14.8

One respondent stated that, in the context of integrating software units into ecosystems, they could replace this interoperability analysis with some negotiation on business rules and software interfaces. This shows a business-oriented analysis for interoperability, which alone cannot reveal the conceptual software mismatches. Another respondent, who also selected R6, reported that in his unit they decided to proceed
integrating units only if they were implemented using same implementation technologies. This indicates a probable awareness problem about the consequences of ignoring the early interoperability analysis. Looking into the demographic characteristics of these 27 participants, we found that the majority were working in small size organizations with less than 50 members (%35) on medium size integration projects (%40) with less than 5 years of integration experience (%61.53). Note, the only correlated factor to this group of participants with statistical significance (i.e., Spearman’s rho ρ = 0.307, p-value = 0.015) was the reported estimation for analysis cost to be rather expensive, which could be what discouraged practitioners from performing it.

**Immature, unstandardized, unsystematic interoperability analysis.** For the rest respondents (N = 37; %57.8), they stated that interoperability analysis actually took place in their integration projects. The responses show that the majority (%67.6) performed it at the beginning of the integration projects and before starting the technical implementation. Though, many other respondents (%24.3) stated that interoperability analysis happened during the technical implementation. Such an approach would obviously require reworking the implemented parts of the integration as long as the analysis shows the existence of more mismatches. Through statistical analysis, we found that the analysis time was significantly correlated to integration experience in terms of number of years (i.e., Spearman’s rho ρ = -0.608, p-value = 0.000) and number of projects (i.e., Spearman’s rho ρ = -0.679, p-value = 0.000). That is, experienced analysts recognized the importance of early analysis. Very few respondents (only %2.7) reported the analysis to happen after the technical implementation, which is the worst time (in terms of rework consequences) to detect mismatches.

The roles responsible for performing the interoperability analysis task varied. However, architects had the biggest presence among the other roles (see Fig. 2). We remarked that unlike the other roles, testers were never reported to take the responsibility for the analysis alone. Some of the respondents chose one role, while mostly (%72.9) they selected at least two roles, which indicates a form of collaboration on the analysis task. In fact, some explicitly reported that collaboration happened between technical managers and engineering team or DevOps. Also, a collaboration between domain experts was reported by one respondent, which would be of great value for analyzing the conceptual level of interoperability. Regarding the team size, %62.2 said it was small (i.e., < 5 members) although the projects’ size ranged from small to large. Unexpectedly, %66.6 of the respondents, who stated that it was performed by exactly one person, worked on large projects.

![Fig. 2 The responsible roles for performing interoperability analysis.](image)

**What?** Digging deeper, we asked the practitioners to specify the types of information they targeted during the interoperability analysis task. The answers showed that technical aspects were dominant. In specific, %75.7 targeted communication constraints (e.g., networking protocols, message formats, etc.) and %64.9 targeted technical syntax (e.g., argument order, data types, etc.). Fewer respondents (on average %58.1) interested in semantic constraints (e.g., terminologies, goals, rationale, etc.) and behavior constraints (e.g., pre-/post- conditions, invariants, interaction protocols, control flow, etc.). Minor shares of attention (on average %48.6) were given to context, structure, and quality. As expected, the covered information aspects were less for those who reported performing interoperability analysis by one person rather than a team. For example, a respondent indicated that the analysis was performed by one developer and the targeted information was the technical communication only. Statistically, the significantly correlated factor to the targeted information during analysis was the responsible role (i.e., Spearman’s rho ρ = 0.493, p-value = 0.002). For example, architects and developers were the main roles who targeted semantic and behavior information.
Further, we found a statistical significance in the correlation between the years of integration experience and targeting the semantic information (i.e., Spearman’s rho $p = 0.333$, p-value = 0.008).

**How?** To better assess the current situation, we asked about the input, process, and output documentation of interoperability analysis. The answers confirmed our expectation that API documentation was the main available input artifact and source of information as stated by $\%7.4$ of the respondents. Other available sources of information included high-level architecture, requirements specification, and source code. However, these apply to white-box integration (e.g., open source software projects) rather than black-box one (e.g., commercial COTs). One respondent considered that contacting the team members of the external software unit was his source of information due to lack of shared artifacts. With respect to the used support for performing the interoperability analysis, we found that about $\%30$ of the practitioners with knowledge about this issue had none. Further, two respondents declared that they did the analysis in an ad hoc way focusing on identifying the gaps between the integration requirements and the offered capabilities by the external software units. On the other hand, $\%13.5$ used analysis models and frameworks, $\%10.8$ followed guidelines, $\%8.1$ performed systematic analysis, $\%8.1$ used a template, and $\%5.4$ had tool support. However, respondents did not give further information, details, or references for the reported support, except for two. One added that the followed guidelines were internally developed and the other added that they had also an internally defined process to follow. With regards to documenting the analysis results, we found that about $\%30$ of the practitioners did not document the results of their interoperability analysis. This documentation status had a statistically significant correlation with integration experience in terms of the number of projects that the participants worked on (i.e., Spearman’s rho $p = -0.438$, p-value = 0.009). Consequently, there was a loss of information that supports decision traceability within a project and loss of knowledge that allows learning from experiences and cases across integration projects.

### 4.3 Problems in Performing Conceptual Interoperability Analysis (RQ2)

Here we present the collected evidence from practitioners on the relevance of the practical problems that are related to interoperability analysis practical support and input artifacts. Then, we shed the light specifically on the problems related to the conceptual level of interoperability analysis. We collected data about perceived problems from all the sample size ($N = 64$). However, for the questions that depend on actual experience with interoperability analysis (i.e., RQ2.1 and RQ2.2), we differentiate between the results obtained from those who claimed to perform the interoperability analysis (actual experiences) and those who did not (opinions).

**High cost of interoperability analysis.** Out of the respondents who had knowledge about the cost ($N = 43$), ($46.51$) reported it ranged from $\%10$ to $\%30$ of the total cost of integration projects and ($20.93$) stated it ranged between $30\%$ and $50\%$. Obviously, such cost is relatively high taking into account the other development activities included within the integration project (e.g., requirement analysis, design, implementation, testing, etc.). In one case, a respondent stated that it would even be worse and would reach $51\%$ to $70\%$ of the total cost. Remarkably, this respondent was one of those who reported performing the analysis during the implementation of the integration. Few amount of responses ($27.91$) indicated that the cost of interoperability analysis would be less than $10\%$. Most of these respondents were of rather low integration experience (i.e., they worked on $2$ to $5$ projects). However, no significant correlations were found between the analysis cost and the demographic features.

**Frequently undetected conceptual interoperability mismatches.** The survey respondents, who had knowledge about the frequency of integration problems related to undetected conceptual mismatches ($N = 59$), stated that it was most likely to happen. The majority ($52.54\%$) stated it happened sometimes, while ($30.51\%$) said it was a usual issue. Only three respondents said that the unexpected conceptual mismatches were always happening. Very few ($N = 7$) claimed that this problem was rare and none said it was never a problem. Statistically, the responses show a significant agreement (i.e., One-Sample Wilcoxon signed-rank $Z = 470$, p-value = 0.000, $H_0$: Median = 3: sometimes) on the frequency of undetected conceptual mismatches.

**Expensive resolution for unexpected conceptual interoperability mismatches.** The respondents, who had knowledge about the resolution cost ($N = 47$), reported rather high additional costs for resolving the undetected mismatches. The majority of respondents ($46.81$) agreed that the total integration project cost
increased by %10 to %30. Furthermore, a considerable portion of them (more than the third) agreed that the added cost would even be 31% to 50%. Note, the few respondents (%6.38) who said that conceptual mismatches were a rare cause of problems, still agreed that they were expensive (e.g., one stated that it would cost > 70%).

4.3.1 Problems Related to Practices of Interoperability Analysis (RQ2.1)

**Perceived need for better practical support for performing interoperability analysis.** We offered the survey participants with a list of practice-related difficulties (D) that would impede performing interoperability analysis and asked them to select what they considered as the main ones. This list included:

- **D1**: Lack of focus on detecting the "conceptual" mismatches compared to the "technical" ones
- **D2**: Lack of support for traceability between interoperability analysis activities and results (i.e., within a project and among projects)
- **D3**: Lack of standard templates for consistent documentation of interoperability analysis results
- **D4**: Lack of interoperability analysis guidelines and best practices for practitioners
- **D5**: Undirected collection of information about the external software units (i.e., no plan or pre-defined data elements)
- **D6**: Posterior collection of information about the external software unit (i.e., reactive collection based on rising problems along the project)
- **D7**: Manual effort in analyzing description of external software units and in documenting the analysis results

According to the responses (N = 64), D4, D7, and D1 had the highest agreement among practitioners as seen in Table 1. This provides evidence on the need for aiding interoperability analysts and architects in identifying the conceptual mismatches with practical guidelines and automation tools. Furthermore, D2 and D3 had also a considerable amount of agreement. Obviously, these two essential difficulties are related, as the ability to trace interoperability analysis results requires documenting them. Accordingly supporting practitioners with standard documentation templates would serve the aforementioned traceability need along with other benefits like consistency and readability. Although D5 and D6 got the least shares of agreements, there were still practitioners who agreed on the importance to overcome them. Hence, directed analysis with proactive preparation can enhance the analysis experience and results for some analysts.

<table>
<thead>
<tr>
<th>Difficulty (D)</th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>D4</th>
<th>D5</th>
<th>D6</th>
<th>D7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total agreements</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>26</td>
<td>17</td>
<td>9</td>
<td>26</td>
</tr>
<tr>
<td>Agreement percentage %</td>
<td>39.06</td>
<td>37.50</td>
<td>35.94</td>
<td>40.63</td>
<td>26.56</td>
<td>14.06</td>
<td>40.63</td>
</tr>
<tr>
<td>Test statistics a</td>
<td>(\chi^2)</td>
<td>.080</td>
<td>.961</td>
<td>3.025</td>
<td>6.723</td>
<td>.225</td>
<td>.337</td>
</tr>
<tr>
<td></td>
<td>(p)</td>
<td>.777</td>
<td>.327</td>
<td>.010*</td>
<td>.635</td>
<td>.562</td>
<td>.618</td>
</tr>
<tr>
<td>Test statistics b</td>
<td>(\rho)</td>
<td>-.035</td>
<td>.123</td>
<td>.217</td>
<td>.324</td>
<td>.059</td>
<td>-.073</td>
</tr>
<tr>
<td></td>
<td>(p)</td>
<td>.781</td>
<td>.335</td>
<td>.084</td>
<td>.009**</td>
<td>.641</td>
<td>.569</td>
</tr>
</tbody>
</table>

| Test statistics a | \(\chi^2\) | .080 | .961 | 3.025 | 6.723 | .225 | .337 | .249 |
| | \(p\) | .777 | .327 | .010\* | .635 | .562 | .618 |
| Test statistics b | \(\rho\) | -.035 | .123 | .217 | .324 | .059 | -.073 | -.062 |
| | \(p\) | .781 | .335 | .084 | .009\** | .641 | .569 | .624 |

\(\chi^2\) test H0: Agreement percentage (respondent who performed interoperability analysis) = Agreement percentage (respondent who did not perform interoperability analysis) = 0

\(\rho\) test H0: There is a correlation between agreements and respondents' group (performed interoperability analysis or not); * \(p < 0.05\); ** \(p < 0.01\); *** \(p < 0.001\)

After a more thorough investigation, we found one statistically significant difference between the agreement percentages on D4 of the two groups of surveyed practitioners (i.e., those who performed interoperability analysis in their software integration projects and those who did not). In fact, this difference was also justified by the statistical significance of the correlation between the group type and the agreement on D4. Although, there were some other percentage differences between the answers per group, however, they were not statistically significant. For example, D2 and D3 had more votes by the inexperienced practitioners in...
performing the analysis task (21.12%, and 12.01% respectively). Thus, we conclude that all reported difficulties are important for both groups, but overcoming D4 would be of higher value for inexperienced practitioners.

4.3.2 Problems related to the Input Artifacts of Interoperability Analysis (RQ2.2)

**Perceived insufficiency of shared information.** According to the respondents with knowledge about the current input artifacts of interoperability analysis (N = 59), they mostly (%37.50) reported it to be “3: Not sufficient”. In other words, on the 5-likert scale, the main rate was 2: insufficient. Statistically, the agreement on insufficiency was significant (i.e., One-Sample Wilcoxon signed-rank Z = -4.76, p-value = 0.000, H0: Median = 4: sufficient) from both groups (i.e., who performed interoperability analysis and who did not).

**Perceived need for enhancing conceptual information content.** The respondents (N = 64) voted for what they perceived as required enhancements for the content of input artifacts used in the interoperability analysis task. We offered a list of interoperability-related content (C) and we asked the respondents to select what they considered as important to enhance the content related to them. This list included the following:

- C1: Communication constraints (e.g., networking protocols, message formats, etc.)
- C2: Syntax constraints (e.g., argument order, data types, etc.)
- C3: Semantic constraints (e.g., glossaries, goals, rationale, etc.)
- C4: High-level architecture view (e.g., architecture style, patterns, etc.)
- C5: Low-level design decisions (e.g., inheritance, synchronicity, concurrency, etc.)
- C6: Behavior constraints (e.g., pre/post conditions, interaction protocols, control flow, etc.)
- C7: Context constraints (e.g., stakeholders, environments, use cases, etc.)
- C8: Quality constraints (e.g., data precision, service performance, etc.)

Based on the responses (N = 64), C4 got the biggest share of practitioners’ interest (see Table 2). This evidently indicates the serious need to enrich the shared documents about interoperable software units with high-level architecture for better analysis. Next, C1 and C6 got substantial agreements, which make them of high priority too. Note that C1 is a technical type of content, while C6 is conceptual. Afterward, C3, C8, C2, and C5 got convergent large shares of respondents’ agreement. This shows the awareness about the need to improve the quality, semantics, syntax and low-level design information of interoperable units. Although C7 got the least share of agreements, it was still agreed on by 24 practitioners as important. These results denote a potential improvement for interoperability analysis results when the content issues are resolved.

<table>
<thead>
<tr>
<th>Content problem (C)</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>C5</th>
<th>C6</th>
<th>C7</th>
<th>C8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total agreements</td>
<td>37</td>
<td>29</td>
<td>31</td>
<td>39</td>
<td>27</td>
<td>36</td>
<td>24</td>
<td>31</td>
</tr>
<tr>
<td>Percentage%</td>
<td>57.81</td>
<td>45.31</td>
<td>48.44</td>
<td>60.94</td>
<td>42.19</td>
<td>56.25</td>
<td>37.50</td>
<td>48.44</td>
</tr>
<tr>
<td>Test statistics a</td>
<td>( \chi^2 )</td>
<td>.098</td>
<td>2.705</td>
<td>2.430</td>
<td>.568</td>
<td>.098</td>
<td>1.246</td>
<td>4.651</td>
</tr>
<tr>
<td></td>
<td>p</td>
<td>.755</td>
<td>.100</td>
<td>.119</td>
<td>.451</td>
<td>.755</td>
<td>.264</td>
<td>.031*</td>
</tr>
<tr>
<td>Test statistics b</td>
<td>p</td>
<td>-.039</td>
<td>-.206</td>
<td>-.195</td>
<td>-.094</td>
<td>.039</td>
<td>-.140</td>
<td>-.270</td>
</tr>
<tr>
<td></td>
<td>p</td>
<td>.759</td>
<td>.103</td>
<td>.123</td>
<td>.459</td>
<td>.759</td>
<td>.272</td>
<td>.031*</td>
</tr>
</tbody>
</table>

* Pearson’s Chi-Square (\( \chi^2 \)) test H0: Agreement percentage (respondent who performed interoperability analysis) = Agreement percentage (respondent who did not perform interoperability analysis)

* Spearman’s rho (\( \rho \)) test H0: There is a correlation between agreements and respondents’ group (performed interoperability analysis or not); * \( p < 0.05 \) ; ** \( p < 0.01 \) ; *** \( p < 0.001 \)

Deeper investigation showed us that there was one statistically significant difference between the agreement percentages on C7 from the two respondent groups (i.e., those who performed interoperability analysis in their software integration projects and those who did not). In addition, there was a statistical significance of the correlation between the group type and the agreement on C7. The percentage differences between the two groups on the content problems not statistically significant. For example, C2 and C3 had more votes by experienced practitioners in performing the analysis task (almost 20% each). We conclude that all content
items are important for both groups, however, experts of interoperability analysis perceived them of higher importance (especially C7) compared to inexperienced respondents.

**Perceived need for enhancing conceptual information presentation.** According to the respondents \( (N = 64) \), there were some required enhancements for the presentation of input artifacts used in the interoperability analysis task. Out of the list of presentation enhancements \( (P) \), which we suggested in the survey, the respondents voted for the ones perceived as the most important. This list included the following:

- **P1**: Mixing conceptual and technical constraints without clear borders between them
- **P2**: Unstructured verbose of text
- **P3**: Lack of easy-to-read process diagrams (e.g., flowcharts)
- **P4**: Inconsistency in reporting constraints for the different data items and services
- **P5**: Too low formality preventing potential automation of analysis

Based on the responses \( (N = 64) \), P1 and P3 got the highest agreements in equal amounts of practitioners’ interest (see Table 3). This points out a critical need to improve the structure of information shared about interoperable software units to clearly differentiate between conceptual and technical information. Moreover, abstract process view could enhance these documents. Afterward, respondents considerably agreed on P4 and P2, which shows that structure and consistency in presenting content among equal elements could improve the usefulness of shared artifacts. Although P5 got the lowest agreement, a large number of practitioners, it was still agreed on by 20 practitioners as important presentation issue. The statistical test results showed that there was one statistically significant difference between the agreement percentages on P3 of the two respondent groups (i.e., those who performed interoperability analysis in their software integration projects and those who did not). Moreover, a statistical significance of the correlation between the group type and the agreement on P3 was found. Hence, we conclude that enhancing the presentation of the content of input artifacts for interoperability analysis would be of value for both experienced and inexperienced analysts. Though, inexperienced ones will appreciate it more, if the processes are abstracted in diagrams rather than in unstructured text.

### Table 3 Perceived need to enhance the presentation of input artifacts for interoperability analysis.

<table>
<thead>
<tr>
<th>Presentation problem (P)</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total agreements</td>
<td>30</td>
<td>25</td>
<td>30</td>
<td>27</td>
<td>20</td>
</tr>
<tr>
<td>Agreement percentage%</td>
<td>46.88</td>
<td>39.06</td>
<td>46.88</td>
<td>42.19</td>
<td>31.25</td>
</tr>
<tr>
<td>Test statistics ( a )</td>
<td>( \chi^2 )</td>
<td>.706</td>
<td>.055</td>
<td>4.854</td>
<td>.040</td>
</tr>
<tr>
<td></td>
<td>( p )</td>
<td>.401</td>
<td>.814</td>
<td>.028*</td>
<td>.841</td>
</tr>
<tr>
<td>Test statistics ( b )</td>
<td>( \rho )</td>
<td>-.105</td>
<td>.029</td>
<td>.275</td>
<td>-.025</td>
</tr>
<tr>
<td></td>
<td>( p )</td>
<td>.409</td>
<td>.818</td>
<td>.028*</td>
<td>.844</td>
</tr>
</tbody>
</table>

* Pearson’s Chi-Square (\( \chi^2 \)) test \( H_0 \): Agreement percentage (respondent who performed interoperability analysis) = Agreement percentage (respondent who did not perform interoperability analysis)

* Spearman’s rho (\( \rho \)) test \( H_0 \): There is a correlation between agreements and respondents’ group (performed interoperability analysis or not); * \( p < 0.05 \); ** \( p < 0.01 \), *** \( p < 0.001 \)

### 5 THREATS TO VALIDITY

**Internal validity (Content validity).** As we described earlier in the survey design (see Section 3.2), we did multiple peer reviews with experts in architecture, software engineering, and empirical research. Further, we evaluated the survey in pilot studies to assess the understandability of the questionnaire.

**External validity (Representative sample).** The final included number of responses is \( (N = 64) \) of software architects and engineers with integration experiences from different organizations, industrial domains, and locations. Thus, we assume our results to be very likely representative for the state of practice of interoperability analysis as of June 2016. However, for better generalization and observations over time, further surveys with larger sample size are required.
External Validity (Completion rate). As peer reviews reported the questionnaire to be long, we shortened it to increase the questions’ completion rate. Also, to collect reliable responses based on genuine experiences, we had conditional appearance of questions and we offered “I don’t know” option for questions seeking knowledge rather than opinions. Thus, we got a completely answered questionnaire by all respondents.

6 SUMMARY AND CONCLUSION

In this paper, we have presented a consolidated description of the current state of practice of software interoperability analysis. We started with describing the as-is situation, which revealed that %30 of practitioners did not perform interoperability analysis in their integration projects. The main reasons behind this were found to be the lack of knowledge about how to perform it and the lack of awareness about its importance, which leads to prioritizing other tasks over it. On the other hand, practitioners who performed interoperability analysis showed us that current state of interoperability analysis was immature. More specifically, there was no standard or systematic activities followed (e.g., some performed interoperability analysis during the implementation rather than before it) and there was no comprehensive investigation for interoperability information during the analysis (e.g., very few of low experienced practitioners targeted conceptual information). Afterward, we put our hands on the exact difficulties related to both the practices and input artifacts of the conceptual interoperability analysis task.

To overcome the difficulties identified in this survey, software engineering researchers should (1) develop rigorous interoperability analysis approaches and automation tools with comprehensive coverage for both conceptual and technical constraints, (2) identify criteria for deciding the cost of resolving the different types of interoperability mismatches, (3) deriving guidelines for improving the content and presentation of shared information about interoperable software unit, and (4) developing guidelines for applying interoperability analysis as well as for standard templates for results to allow reuse experiences and decisions.

ACKNOWLEDGMENTS

This work was funded by the Ph.D. Program and Nachwuchsrings of the University of Kaiserslautern. We thank, Liliana Guzman and Matthias Naab for reviewing the study design and the questionnaire.

REFERENCES