Specification Patterns from Research to Industry: A Case Study in Service-Based Applications

Domenico Bianculli
Faculty of Informatics
University of Lugano
Lugano, Switzerland
domenico.bianculli@usi.ch

Carlo Ghezzi
DEEPSE group - DEI
Politecnico di Milano
Milano, Italy
ghezzi@elet.polimi.it

Cesare Pautasso
Faculty of Informatics
University of Lugano
Lugano, Switzerland
cesare.pautasso@usi.ch

Patrick Senti
Information Technology
Credit Suisse AG
Zürich, Switzerland
patrick.senti@credit-suisse.com

***Abstract*—Specification patterns have proven to help developers to state precise system requirements, as well as formalize them by means of dedicated specification languages. Most of the past work has focused its applicability area to the specification of concurrent and real-time systems, and has been limited to a research setting. In this paper we present the results of our study on specification patterns for service-based applications (SBAs). The study focuses on industrial SBAs in the banking domain. We started by performing an extensive analysis of the usage of specification patterns in published research case studies — representing almost ten years of research in the area of specification, verification, and validation of SBAs. We then compared these patterns with a large body of specifications written by our industrial partner over a similar time period. The paper discusses the outcome of this comparison, indicating that some needs of the industry, especially in the area of requirements specification languages, are not fully met by current software engineering research.**

***Keywords*-specification patterns; specification languages; requirements specifications; services**

I. Introduction

The concept of *pattern* has been initially proposed in the domain of architecture by C. Alexander [1], to represent "the description of a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice".

This idea of pattern has then been adopted in software engineering with the concept of *design patterns* [2], as reusable solutions for recurring problems in software design. Subsequently, the concept of design patterns has been embraced in different sub-domains of software engineering, from architectural patterns to reengineering patterns, including property specification patterns.

Property specification patterns [3] have been proposed in the late '90s in the context of finite-state verification, as a means to express recurring properties in a generalized form, which could be formalized in different specification languages, such as temporal logic. Specification patterns aimed at bridging the gap between finite-state verification tools (e.g., model checkers) and practitioners, by providing the latter with a powerful instrument for writing down properties to be fed to a formal verification tool.

Given the origin of property specification patterns, most of past work has focused its applicability area to the specification (and the verification) of concurrent and real-time systems (see, for example, [4]), with limited applications outside the research setting.

In the last years, open software [5] systems such as service-based applications (SBAs) have emerged, introducing new engineering challenges due to their dynamic and decentralized nature. One of these research challenges is related to the specification, verification and validation of SBAs [6]. At the same time, service-oriented architectures (SOAs) have gained a lot of attention in enterprises, which started to adopt them for the integration of their information systems [7]. However, to the best of our knowledge, the research literature on specification, verification and validation of SBAs has presented limited evidence of its applicability to and suitability for industrial-level case studies.

One of the questions that we asked ourselves during our research is whether existing requirements specification languages are expressive enough to formalize common requirements specifications used in industry. In particular, we are interested in evaluating the use of specification patterns for expressing properties of industrial SBAs, to assess if existing and well-known specification patterns are adequate or not. If this is not the case, our goal is to gather substantial evidence for new specification patterns and/or language constructs required to support their practical use in industrial settings.

In this paper we present the results of our study on the use of specification patterns in SBAs. The study has been performed by analyzing the requirements specifications of two sets of case studies. One set was composed of case studies extracted from research papers in the area of specification, verification and validation of SBAs, which appeared in the main publishing venues of software engineering and service-oriented computing within the last 10 years. The other set was composed of case studies corresponding to

service interfaces written over a similar time period and used within the SBAs developed by our industrial partner, which operates in the banking domain.

During the analysis, we matched each SBA requirements specification against the patterns belonging to the specification pattern systems we selected from the research literature. When a match was not possible, we tried to classify the requirements specification according to a new pattern system, specific to the service provisioning domain, that we had been building during the matching process. Finally, we compared the results, in terms of matched patterns, for the research and the industrial case studies.

Anticipating the results of the study, which are presented in detail in the paper, we can state that the comparison of the patterns usage between the research and the industrial case studies shows that: a) the majority of requirements specifications stated in industrial settings refers to specific aspects of service provisioning, which can be characterized as a new class of specification patterns; b) the specification patterns proposed in the research literature are barely used in industrial settings.

These considerations indicate that some needs of the industry are not fully met by software engineering research. This suggests that new research directions in the areas of requirements specification languages and of the related verification techniques should be explored. The outcome of this research has the potential of making it possible to provide industry with a holistic solution, made of suitable specification language(s) and related verification tools, to be integrated into the quality assurance process of SBAs.

The rest of this paper is organized as follows. Section II illustrates the specification patterns considered in this survey. Section III describes the methodology used to conduct the survey and presents its results, which are then discussed in Section IV. Related work is presented in Section V, while Section VI concludes the paper.

II. A Bird's Eye View of Specification Patterns

In this section we summarize the patterns we have used to classify the surveyed service specifications. We have grouped them in four classes: the first three groups correspond to systems of specification patterns well-known in the software engineering research community, but not necessarily used in the context of SBAs, while the last one includes patterns that are more specific to service provisioning. For each pattern we include a brief description as well as a simple property expressed using the pattern; in the sample properties, we use the letters P, S, T, and Z to denote events or states of a system execution.

The "D" Group

The first group corresponds to the property specification pattern system originally proposed by Dwyer et al. in [3]. This system includes nine parameterizable, high-level, formal specification abstractions. These patterns can be combined with five scopes ("global", "before", "after", "between", and "after until"), to indicate the portions of a system execution in which a certain pattern should hold. Note that in the rest of the paper, we do not distinguish among the different scopes with which a certain pattern has been used, and report usage data aggregated over all possible scopes. The patterns are[1]:

Absence (D1) describes a portion of a system's execution that is free of certain events or states, as in "it is never the case that P holds".

Universality (D2) describes a portion of a system's execution that contains only states that have a desired property, as in "it is always the case that P holds".

Existence (D3) describes a portion of a system's execution that contains an instance of certain events or states, as in "P eventually holds".

Bounded existence (D4) describes a portion of a system's execution that contains at most a specified number of instances of a designated state transition or event, as in "it is always the case that the transitions to state P occur at most 2 times".

Precedence (D5) describes relationships between a pair of events/states, where the occurrence of the first is a necessary pre-condition for an occurrence of the second, as in "it is always the case that if P holds, then S previously held".

Response (D6) describes cause-effect relationships between a pair of events/states, where an occurrence of the first must be followed by an occurrence of the second, as in "it is always the case that if P holds, then S eventually holds".

Response chains (D7) is a generalization of the response pattern, as it describes relationships between *sequences* of individual state/events, as in "it is always the case that if P holds, and is succeeded by S, then T eventually holds after S".

Precedence chains (D8) is a generalization of the precedence pattern, as it describes relationships between *sequences* of individual state/events, as in "it is always the case that if P holds, then S previously held and was preceded by T".

Constrained chain patterns (D9) describes a variant of response and precedence chain patterns that restricts user specified events from occurring between pairs of states/events in the chain sequences, as in "it is always the case that if P holds, then S eventually holds and is succeeded by T where Z does not hold between S and T".

The "R" Group

The second group of patterns has been proposed by Konrad and Cheng [8] in the context of real-time specifications.

[1]A detailed description is available at http://patterns.projects.cis.ksu.edu.

This pattern system includes five patterns (and the same five scopes as in [9]) as well as a structured English grammar that supports both qualitative and real-time specification patterns. The five patterns are:

Minimum duration (R1) indicates the minimum amount of time a state formula has to hold once it becomes true, as in "it is always the case that once P becomes satisfied, it holds for at least *k* time units".

Maximum duration (R2) describes that a state formula always holds for less than a specified amount of time, as in "it is always the case that once P becomes satisfied, it holds for less than *k* time units".

Bounded recurrence (R3) indicates the amount of time in which a state formula has to hold at least once, as in "it is always the case that P holds at least every *k* time units".

Bounded response (R4) indicates the maximum amount of time that passes after a state formula holds until another state formula becomes true, as in "it is always the case that if P holds, then S holds after at most *k* time units".

Bounded invariance (R5) indicates the minimum amount of time a state formula must hold once another state formula is satisfied, as in "it is always the case that if P holds, then S holds for at least *k* time units".

The "G" Group

Another system of real-time specification patterns was developed, around the same time as the previous one, by Gruhn and Laue [10]. The system includes the actual patterns, certain types of combined events that can be used within specifications, and scopes that determine patterns validity. As for scopes, the authors support the possibility to express that a property holds before, after, and until a certain number of time units (possibly zero) have passed since the last occurrence of a certain event. The patterns are:

Time-bounded existence (G1) is the timed version of pattern D3, meaning that it can express properties such as "starting from the current point of time, P must occur within *k* time-units".

Time-bounded response (G2) represents the same pattern as R4.

Precedence with delay (G3) represents, together with the next pattern, the timed version of pattern D5. In this first variant, it can state properties such as "P must always be preceded by S and at least *k* time units have passed since the occurrence of S".

Time-restricted precedence (G4) is the second timed variant of pattern D5; it can express properties such as "P must always be preceded by S and must occur within at most *k* time units since the occurrence of S".

The "S" Group

This group combines the patterns we found in the literature dealing with SBAs specifications, which do not appear in the pattern systems described above; for this reason, we group them all together under *service provisioning* patterns.

Average response time (S1) is a variant of the bounded response pattern (R4) that uses the average operator to aggregate the response time over a certain time window.

Counting the number of events (S2) is used (see, for example, [11]) to express common non-functional requirements such as reliability (e.g., "number of errors in a given time window") and throughput (e.g., "number of requests that a client is allowed to submit in a given time window").

Average number of events (S3) is a variant of the previous pattern that states the average number of events occurred in a certain time interval within a certain time window, as in "the average number of client requests per hour computed over the daily business hours".

Maximum number of events (S4) is another variant of the S2 pattern that aggregates events using the maximum operator, as in "the maximum number of client requests per hour computed over the daily business hours".

Absolute time (S5) indicates events that should occur at a time that satisfies an absolute time constraint, as in "if the booking is done in the first week of March, a discount is given" (taken from [12]).

Elapsed time (S6) indicates the time elapsed since the last occurrence of a certain event.

Data-awareness (S7) is a pattern denoting properties that refer to the actual data content of messages exchanged between services as in "every ID present in a message cannot appear in any future message" (taken from [13]).

III. The Survey

In our study, we extracted specification patterns for SBAs by analyzing examples and case studies both from the research literature and from industry.

We analyzed the requirements specifications for the SBA(s) described in each example or case study, and manually classified each specification to match the patterns defined in the previous section. The specifications were in many forms: some were expressed using a specification formalism (e.g., a temporal logic), while others were expressed in the English natural language. When a specification could not be easily matched with a pattern, we used the criteria proposed in [9] to still count a specification as a match: a) formal equivalence; b) equivalence by parameter substitution; c) variant of a pattern; d) wrong formal specification with matching prose description.

Note that a single requirements specification may match more than one pattern; for example, a requirement such as "*if a message with a red code alert is received three times for the same patient during a time span of a week, then doctors should send a confirmation for the hospitalization of that patient within a hour from the reception of the last alert message*" is an instantiation of patterns R4 (*bounded response time*), S2 (*counting*) and S7 (*data-awareness*).

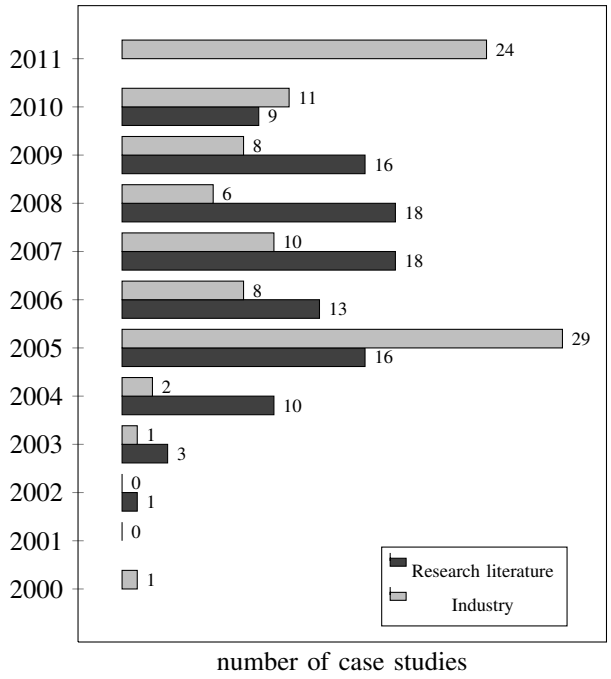

Figure 1. Number of case studies considered per year

The set of case studies we considered spans over more than ten years, as shown in Figure 1. Overall, we considered 104 case studies from the research literature and 100 industrial ones.

In the rest of this section we describe, for each of the two categories of case studies, the data sources and the data themselves.

A. Research Literature Data

The research case studies have been extracted from papers[2] published between 2002 and 2010; the reason for choosing 2002 as the left bound is that research in the area of (Web) SBAs originated around that time. As publication venues to analyze, we considered the main conferences in software engineering (ASE, FASE, SIGSOFT FSE, ICSE), the main conferences in service-oriented computing (ECOWS, ICSOC, ICWS, SCC, SERVICES, SOCA, WSFM, WWW), the major journals in the two areas (respectively, ACM TOSEM and IEEE TSE for software engineering, and ACM TWEB and IEEE TSC for service-oriented computing). For each of these venues, we selected papers on specification, validation, and verification of SBAs; from this set, subsequently, we only considered papers with at least one case study with at least one requirements specification[3].

[2]The complete bibliography of papers considered in this study is available in PDF and BibTeX format on the authors' web site.

[3]In few cases, we also considered papers that included at least one requirements specification formulated in a general way, i.e., not related to a specific example or case study.

Table I
NUMBER OF PAPERS CONSIDERED, PER SCIENTIFIC VENUE, PER YEAR

venue	2002	2003	2004	2005	2006	2007	2008	2009	2010
ASE	0	1	1	0	0	0	0	0	0
ECOWS	–	0	0	1	0	2	0	2	1
FASE	0	0	0	1	0	0	1	0	0
FSE	0	0	0	0	0	0	1	1	1
ICSE	0	0	1	0	0	0	1	0	0
ICSOC	–	0	3	3	0	2	1	1	1
ICWS	–	0	1	1	4	3	2	2	1
SCC	–	–	0	1	0	1	1	1	0
SERVICES	–	–	–	–	–	–	0	0	0
SOCA	–	–	–	–	–	1	–	0	0
WSFM	–	–	1	1	2	2	2	1	0
WWW	0	0	1	1	1	0	0	0	0
TOSEM	0	0	0	0	0	0	0	0	1
TSE	0	0	0	0	0	0	0	2	1
TWEB	–	–	–	–	–	0	0	0	1
TSC	–	–	–	–	–	–	0	2	0
other	1	2	2	5	8	7	5	3	2
total	1	3	10	14	15	18	14	15	9

Moreover, we also included other papers on specification, verification, and verification of SBAs that we were aware of and that had appeared at other venues; however, these venues have not been systematically surveyed. An overview of the number of papers considered, for each venue, is shown in Table I; note that the values displayed in the table on the row labeled "total" do not match the values shown in Figure 1 because in some cases the same paper illustrated more than one case study.

Although we analyzed 104 case studies, we counted only 36 distinct examples, i.e., in many cases, the same example has been used in different case studies across various papers. The top four recurring examples are "loan approval" (13 times), "travel agency" (12 times), "online shopping" (11 times), and "car rental" (8 times).

Out of these case studies, we analyzed and classified 290 requirements specifications. We successfully matched 272 specifications against the patterns presented in Section II; the pattern usage distribution is shown in Table II.

A portion of these data (the group corresponding to patterns D1–D8, representing the 63% of the specifications) can be compared with existing data available in literature. Indeed, reference [9] presents the usage frequency for patterns in the "D" group, extracted from a set of 511 matched specifications belonging to various application domains, such as hardware and communication protocols, control systems, and distributed object systems. The comparison of our data about patterns D1–D8, with respect to the data presented in [9] is shown in Figure 2. Despite different rankings, the five most common patterns are the same (D1, D2, D3, D5, and D6); moreover, the most common pattern is D6 (*response*), with a similar usage frequency in both data sets.

Table II
PATTERNS USAGE IN RESEARCH SPECIFICATIONS

pattern	occurrence	distribution %
D6	76	27.9
R4	52	19.1
S7	47	17.3
D5	22	8.1
D1	20	7.4
S2	20	7.4
D2	19	7
D3	17	6.3
D7	8	2.9
D8	6	2.2
S1	5	1.8
D4	3	1.1
G1	2	0.7
S3	2	0.7
S5	2	0.7
S6	2	0.7
R3	1	0.4
G3	1	0.4
D9	0	0
R1	0	0
R2	0	0
R5	0	0
G4	0	0
S4	0	0

Table III
PATTERNS USAGE IN INDUSTRIAL SPECIFICATIONS

pattern	occurrence	distribution %
S3	201	35.8
S4	168	29.9
S7	97	17.3
S1	91	16.2
D6	11	2
D1	1	0.2
D2	0	0
D3	0	0
D4	0	0
D5	0	0
D7	0	0
D8	0	0
D9	0	0
R1	0	0
R2	0	0
R3	0	0
R4	0	0
R5	0	0
G1	0	0
G3	0	0
G4	0	0
S2	0	0
S5	0	0
S6	0	0

B. Industrial Data

The industrial case studies have been provided by our industrial partner Credit Suisse, a world-leading financial services company headquartered in Switzerland.

Credit Suisse started to implement an SOA in 2000, as a means to leverage its encompassing set of "legacy" mainframe IT applications. In the process, Credit Suisse has established one of the largest CORBA-based service backbones in industry, which has recently been extended to support Web services standards [14]. Credit Suisse operates the Interface Management System (IFMS) as a central information base for all service interfaces available for reuse [15]. IFMS is an integral part of an application developer's work process: not only it does provide documentation on interfaces, but it also generates the required code artifacts (service stubs) to use a service. For new service interfaces, IFMS provides a workflow covering all tasks related to definition, specification, and quality management, thus linking involved staff during the phases of service development, testing and deployment.

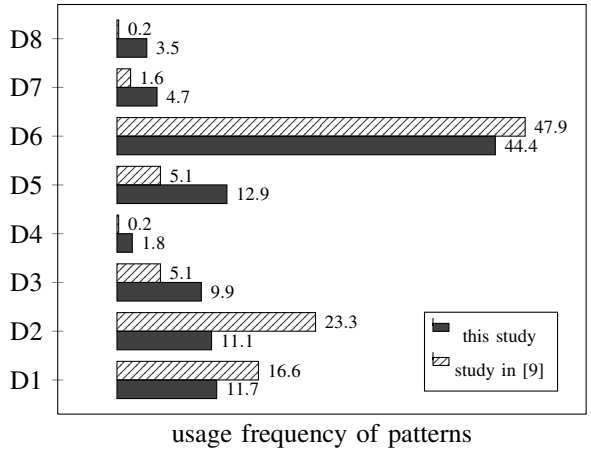

Figure 2. Comparison of the usage frequency of patterns of the "D" group in research case studies, as reported by our study and by reference [9]

The service specifications analyzed in this paper were extracted from IFMS by selecting a random subset of 100 service interfaces. They cover the whole range of application domains at Credit Suisse, such as accounts, payments, customers, financial securities operations, and stock exchange. When an interface contained multiple versions of a service, we extracted specifications from the most recent version. The selected interfaces have been defined between 2000 and 2011.

The general structure of an interface document includes, among others, sections about pre- and post-conditions of the service, as well as on non-functional assertions under different usage conditions; we extracted requirement specifications from all these sections, when available.

In total, we extracted 625 requirements specifications from the set of 100 case studies. We matched 562 of them against the patterns presented in Section II; the pattern usage distribution is shown in Table III.

IV. Discussion

To compare them, the pattern usage distributions of Tables II and III have been combined and plotted on the chart displayed in Figure 3. It is possible to immediately see the discrepancy of pattern usage between research and industrial case studies, with a separation line virtually drawn before the patterns of the "S" group.

It is clear that the majority of requirements specifications used in industrial settings matches the S1, S3, S4 and S7 patterns; below, we discuss the usage of each of these patterns in the two categories.

As for pattern S1 ("average response time"), we have already stated that it can be considered a variant of pattern R4 ("response time"); moreover, R4 is the second most used pattern in the specifications from the research literature. In light of this, we can compare the usage of pattern S1 (16.2%) in industrial specifications with the combined usage of patterns S1 and R4 (20.9%) in the ones from research literature; furthermore, we should also consider that pattern R4 is not used at all in industrial specifications. It is evident that the concept of response time has the same importance, in terms of relevance for the specifications, in both categories of specifications. However, while this concept is used exclusively in its aggregated form (through the average operator) in the industrial specifications, this is not true for research case studies, where the aggregate variant has been used only in five properties (found across five papers).

Similar observations can be made by comparing the usage of patterns S3 and S4 (respectively, "average" and "maximum number of events"), since they represent aggregated variants of pattern S2 ("counting the number of events"). As for the other pattern considered above, it is evident that industrial specifications use only aggregated variants (through the average and maximum operators) of the concept represented by pattern S2. Moreover, aggregated variants of pattern S2 are used very rarely in research case studies; in this case, only pattern S3 is used, and only in two properties (across two papers, from the same authors). Another point to consider is that while counting patterns such as S3 and S4 represent the majority (65.7%) of industrial specifications, the combined usage of patterns S2 and S3 in research specifications is only 8.1%.

As for pattern S7 ("data-awareness"), the figure (17.3%) of its usage in both set of case studies is the same. Indeed, we have noticed in both sets of specifications that this pattern is often used to state pre-/post-conditions on data exchanged by a service. We also note that some recent research (for example, see [13]) has investigated support for data-aware properties in specification languages such as temporal logics, representing a good example of an industrial need met by academic research.

Finally, the remaining patterns matched by industrial specifications have been D1 ("absence"), matched only once, and D6 ("response"), matched eleven times. These two patterns actually represent the only patterns matched from the "D", "R" and "G" groups within the set of industrial case studies.

All the observations made above imply two main points:

- The majority of requirements specifications stated in industrial settings refers to non-functional properties expressed using aggregate operators (e.g., average, count, maximum). Similar requirements are found only rarely in the research literature and when so, they are expressed using the non-aggregated versions of the patterns.
- The specification patterns proposed in the research literature are barely used in industrial settings. This may be an indication of the lack of need for expressing such properties within industrial specifications.

Addressing these points requires additional actions.

As for the first point, we believe this study shows some needs of the software industry that should be addressed by the research community. In particular, we see the need for requirements specification languages and verification techniques and tools that support non-functional properties expressed using aggregate operators, since they seem to play an important role in the quality assurance process of industrial systems. We should point out some recent work in this direction, such as support for aggregates in first-order logic [16] and in a modal logic [17], as well as support for quantitative properties in model checking of temporal logics [18] and run-time monitoring of service-level agreements [11] based on timed automata.

For the second point, we plan to conduct a future study involving a group of requirements engineers and developers to answer questions like:

- Do you know about the existence of property specification patterns?
- Which property specification patterns are you familiar with?
- Assuming you know specification patterns, would you use them to write requirements specifications? Why?
- Would specification patterns help you to better understand specifications and avoid ambiguity?

Threats to Validity

The validity of the results presented in this study may be affected by several threats. First, the case studies we have considered from the published research literature may not be adequate representatives of research being developed in the domain of SBAs. Other studies could consider different scientific venues and maybe even extract specifications from case studies presented in different research sub-areas, such as service discovery and dynamic service composition.

Similarly, another threat is represented by the fact that we have analyzed the specifications of case studies provided by

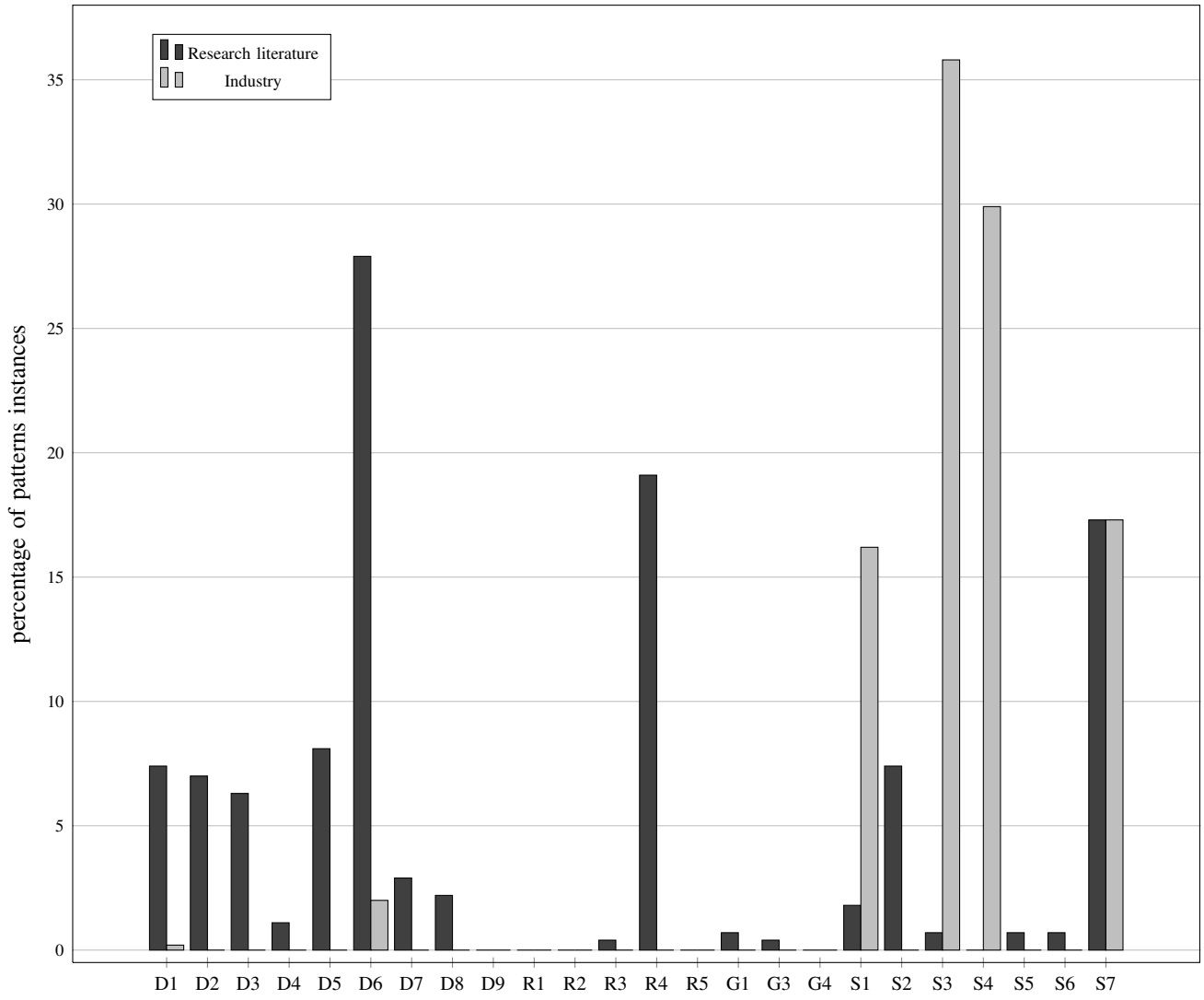

Figure 3. Comparison of patterns usage (percentage) between the research and the industrial case studies

a single industrial organization. Other industries adopting SOAs could define different requirements for their services. Thus, the results obtained so far could be broadened with a survey involving multiple industrial partners.

Finally, the matching of specifications with patterns has been performed manually by a single person with six years of experience in the areas of formal specification and verification, as well as service-oriented computing. Other people could classify the specifications differently, especially when the matching with known patterns is not trivial. Furthermore, given that a certain percentage (10% in the case of research literature data, 20% for the industrial ones) of the requirements specifications were expressed using natural language, a certain degree of intrinsic ambiguity is involved in the interpretation of the properties for the purpose of their classification.

V. Related Work

Although in this paper we have considered only three systems of specification patterns, other similar systems have been presented in the literature. Below, we briefly summarize the pattern systems we did not consider for this study and explain why we opted for the ones presented in Section II. In the area of qualitative temporal specifications, a catalogue of safety patterns is presented in [19]; however, with respect to the "D" patterns group, it is restricted only to safety patterns occurring in the specification of industrial automation systems. Other extensions of the "D" patterns are proposed in [20], which deals with the support of events in LTL formulae, and in [21], where the PROPEL approach — based on a "disciplined" natural language and finite state automata — is used to express fine-tuned versions of the "D" patterns. Since in our case studies we wanted to assess

the usage of the "D" patterns at a high level, we did not go for such more specialized versions of these patterns. As for the area of real-time specifications, a system of patterns using structured English sentences is described in [22]; however, this work is tailored for clocked computational tree logic, while we wanted to use specification language-agnostic pattern systems, such as the "R" and "G" pattern groups.
Another class of specification patterns we did not include in this study is represented by patterns for probabilistic quality properties [23]. For the sake of completeness, we should say that three requirements specifications from the set of research case studies were actual matches for two of the patterns introduced in [23], while none of the specifications of the set of industrial case studies was suitable to be expressed using a probabilistic property pattern.

This paper is also one of the few that reports quantitative data on the usage of certain specification pattern systems in practical examples. Similar data can also be found in [9], as shown in Section III-A; in [8], though the usage distribution of each pattern is not actually disclosed; in [23], for probabilistic property patterns; in [24], which presents a study, conceptually similar to ours, on the analysis of the usage of the "R" patterns in the automotive domain.

The "D" group is also at the basis of some work that focuses on the specification and verification of service interactions in SBAs. For example, in [25], property patterns are defined in an ontology, whose concepts can then be used by developers to describe the interaction behavior of services as constraints. These constraints specify the occurrence and sequencing rules of service invocations and are checked at run time by a dedicated monitoring infrastructure. A similar approach is also followed in [26], where service conversations are specified using a subset of UML 2.0 Sequence Diagrams, which are shown to be able to express all the "D" patterns. Reference [27] presents PROPOLS, a specification language based on the "D" patterns, which adds support for the logic composition of patterns; this language can be used to describe some properties against which service composition workflows can be checked for compliance with a static verification tool. Another specification language, PL, also based on the "D" pattern group, is presented in [28]; the language is used to express behavioral properties of business processes, which can then be automatically translated into a process algebra for refinement checking.

Other work has defined specification languages for service interactions based on real-time patterns, as for the case of the XTUS-Automata language proposed in [12], which also presents the companion run-time monitoring infrastructure. This work presents two additional patterns, "temporal properties over cardinalities" and "absolute time properties", which match, respectively, patterns S2 and S5 in Section II.

VI. Conclusions and Next Steps

In this paper we have presented the results of our comparative study on the use of specification patterns in service-based applications. We compared the usage of patterns for the requirements specifications of research and industrial case studies, gathered over a time period of more than 10 years. The results show that the industrial case studies tend not to use the specification patterns proposed in the research literature, in favor of other patterns that characterize specific aspects of service provisioning and that, conversely, are not common in research case studies.

As part of future work, we plan to conduct a study with a group of requirements engineers and developers of our industrial partner, to assess a) to which extent specification patterns are known in industrial settings; b) if and how much the use of specification patterns helps to better express and understand requirements. For the latter point, we also intend to work with engineers and developers to analyze the requirements specifications we could not match against a pattern (as described in Section III-B), and try to reformulate the requirements using some known pattern(s). Furthermore, we will look at other non-functional requirements, such as the ones related to security and privacy, and see either if they match the security patterns already presented in literature [29] or if new patterns are needed for this specific domain. Last, we will use the outcome of this study to guide the future development of ALBERT, the specification language for service interactions we presented in [30], so that we will be able to apply it and its companion verification methodology in the context of industrial case studies.

Acknowledgment

This work has been partially supported by the European Community under the grant agreement no. EU-FP7-215483-S-Cube and the IDEAS-ERC grant agreement no. 227977-SMScom; by the Swiss NSF under the grant agreement no. 135051-CLAVOS.

References

[1] C. Alexander, S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King, and S. Angel, *A pattern language. Towns, buildings, construction*. Oxford University Press, 1977.

[2] E. Gamma, R. Helm, R. E. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[3] M. B. Dwyer, G. S. Avrunin, and J. C. Corbett, "Property specification patterns for finite-state verification," in *FMSP '98: Proceedings of the 2nd workshop on Formal methods in software practice*. ACM, 1998, pp. 7–15.

[4] J. C. Corbett, M. B. Dwyer, J. Hatcliff, S. Laubach, C. S. Păsăreanu, Robby, and H. Zheng, "Bandera: extracting finite-state models from Java source code," in *ICSE 2000: Proceedings of the 22nd International Conference on Software Engineering*. ACM, 2000, pp. 439–448.

[5] L. Baresi, E. Di Nitto, and C. Ghezzi, "Toward open-world software: Issue and challenges," *IEEE Computer*, vol. 39, no. 10, pp. 36–43, 2006.

[6] L. Baresi and E. Di Nitto, Eds., *Test and Analysis of Web Services*. Springer, 2007.

[7] N. Josuttis, *SOA in Practice: The Art of Distributed System Design*. O'Reilly Media, Inc., 2007.

[8] S. Konrad and B. H. C. Cheng, "Real-time specification patterns," in *ICSE '05: Proceedings of the 27th International Conference on Software Engineering*. ACM, 2005, pp. 372–381.

[9] M. B. Dwyer, G. S. Avrunin, and J. C. Corbett, "Patterns in property specifications for finite-state verification," in *ICSE '99: Proceedings of the 1999 International Conference on Software Engineering*. IEEE Computer Society Press, 1999, pp. 411–420.

[10] V. Gruhn and R. Laue, "Patterns for timed property specifications," *Electron. Notes Theor. Comput. Sci.*, vol. 153, no. 2, pp. 117–133, 2006.

[11] F. Raimondi, J. Skene, and W. Emmerich, "Efficient online monitoring of web-service SLAs," in *SIGSOFT FSE 2008: Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 2008, pp. 170–180.

[12] S. Kallel, A. Charfi, T. Dinkelaker, M. Mezini, and M. Jmaiel, "Specifying and monitoring temporal properties in web services compositions," in *ECOWS 2009: Proceedings of the 7th European Conference on Web Services*. IEEE Computer Society, 2009, pp. 148–157.

[13] S. Halle, R. Villemaire, and O. Cherkaoui, "Specifying and validating data-aware temporal web service properties," *IEEE Trans. Softw. Eng.*, vol. 35, no. 5, pp. 669–683, 2009.

[14] D. Krafzig, K. Banke, and D. Slama, *Enterprise SOA: Service-Oriented Architecture Best Practices*. Prentice Hall, 2004.

[15] S. Murer and B. Bonati, *Managed Evolution: A Strategy for Very Large Information Systems*. Springer, 2010.

[16] N. Pelov, M. Denecker, and M. Bruynooghe, "Well-founded and stable semantics of logic programs with aggregates," *Theory and Practice of Logic Programming*, vol. 7, no. 3, pp. 301–353, May 2007.

[17] C. Areces, G. Hoffmann, and A. Denis, "Modal logics with counting," in *WoLLIC 2010: Proceedings of the 17th Workshop on Logic, Language, Information and Computation*, 2010.

[18] U. Boker, K. Chatterjee, T. A. Henzinger, and O. Kupferman, "Temporal specifications with accumulative values," in *LICS'11: Proceedings of the 26th Symposium on Logic in Computer Science*, 2011.

[19] F. Bitsch, "Safety patterns — the key to formal specification of safety requirements," in *SAFECOMP 2001: Proceedings of the 20th International Conference on Computer Safety, Reliability and Security*, ser. LNCS, vol. 2187. Springer, 2001, pp. 176–189.

[20] M. Chechik and D. O. Paun, "Events in property patterns," in *SPIN 1999: Proceedings of the 5th and 6th International SPIN Workshops on Theoretical and Practical Aspects of SPIN Model Checking*. Springer, 1999, pp. 154–167.

[21] R. L. Smith, G. S. Avrunin, L. A. Clarke, and L. J. Osterweil, "Propel: an approach supporting property elucidation," in *ICSE 2002: Proceedings of the 22rd International Conference on Software Engineering*. ACM, 2002, pp. 11–21.

[22] S. Flake, W. Müller, and J. Ruf, "Structured english for model checking specification," in *Trans. Amer. Math. Soc*. VDE Verlag, 2000, pp. 2547–2552.

[23] L. Grunske, "Specification patterns for probabilistic quality properties," in *ICSE 2008: Proceedings of the 30th International Conference on Software Engineering*. ACM, 2008, pp. 31–40.

[24] A. Post, I. Menzel, and A. Podelski, "Applying restricted English grammar on automotive requirements — does it work? a case study," in *REFQS 2011: Proceedings of the 17th International Working Conference on Requirements Engineering: Foundation for Software Quality*, ser. LNCS, vol. 6606. Springer, 2011, pp. 166–180.

[25] Z. Li, J. Han, and Y. Jin, "Pattern-based specification and validation of web services interaction properties," in *ICSOC 2005: Proceedings of the 3rd International Conference on Service-oriented computing*, ser. LNCS, vol. 3826. Springer, 2005, pp. 73–86.

[26] J. Simmonds, M. Chechik, S. Nejati, E. Litani, and B. O'Farrell, "Property patterns for runtime monitoring of web service conversations," in *RV 2008: 8th International Workshop on Runtime Verification*, ser. LNCS, vol. 5289. Springer, 2008, pp. 137–157.

[27] J. Yu, T. Manh, J. Han, Y. Jin, Y. Han, and J. Wang, "Pattern based property specification and verification for service composition," in *WISE 2006: Proceedings of the 7th International Conference on Web Information Systems Engineering*, ser. LNCS, vol. 4255. Springer, 2006, pp. 156–168.

[28] P. Wong and J. Gibbons, "Property specifications for workflow modelling," in *IFM 2009: Proceedings of the 7th International Conference on Integrated Formal Methods*, ser. LNCS, vol. 5423. Springer, 2009, pp. 56–71.

[29] G. Spanoudakis, C. Kloukinas, and K. Androutsopoulos, "Towards security monitoring patterns," in *SAC '07: Proceedings of the 2007 ACM Symposium on Applied Computing*. ACM, 2007, pp. 1518–1525.

[30] L. Baresi, D. Bianculli, C. Ghezzi, S. Guinea, and P. Spoletini, "Validation of web service compositions," *IET Softw.*, vol. 1, no. 6, pp. 219–232, 2007.