Preparing for a Literature Survey of Software Architecture using Formal Concept Analysis

Luís Couto and José Nuno Oliveira
CCTC
Departamento de Informática
Universidade do Minho
Braga, Portugal
{*pg15260@alunos.uminho.pt, jno@di.uminho.pt*}

Miguel Ferreira and Eric Bouwers
Software Improvement Group
Amsterdam, Netherlands
{*m.ferreira, e.bouwers*}*@sig.eu*

***Abstract*—The scientific literature on Software Architecture (SA) is extensive and dense. With no preparation, surveying this literature can be a daunting task for novices in the field. This paper resorts to the technique of Formal Concept Analysis (FCA) in organizing and structuring such a body of knowledge.**

We start by surveying a set of 38 papers bearing in mind the following questions: "*What are the most supported definitions of software architecture?*", "*What are the most popular research topics in software architecture?*", "*What are the most relevant quality attributes of a software architecture?*" and "*What are the topics that researchers point out as being more interesting to explore in the future?*". To answer these questions we classify each paper with appropriate keywords and apply FCA to such a classification. FCA allows us to structure our survey in the form of *lattices* of concepts which give evidence of main relationships involved.

We believe our results will help in guiding a more comprehensive, in-depth study of the field, to be carried out in the future.

***Keywords*-Introductory and Survey; Software Architectures; Formal Concept Analysis;**

I. Introduction

Architecture is a relevant aspect of software systems because it "*allow[s] or preclude[s] nearly all of the systems quality attributes*" [38]. Although the term "architecture", referring to software, seems to be widely accepted in both academia and industry we notice that there is no consensual understanding of it. A quick overview of just a few papers will reveal significantly different definitions of SA, for instance. This calls for knowledge classification and aggregation in the field. Our main motivation in this paper is to use formal concept analysis (FCA) [39] to help us in classifying and structuring the vast bibliography in the area, paving the way to the future construction of a taxonomical body of knowledge. We consider that a survey could help to clear this and other issues, such as what the main topics of the field are, what quality attributes are more relevant to consider when working with the architecture of a software system, or what the most promising topics for future research are. Before starting a full fledge literature review of the field (as proposed in [40] for instance), we propose to do a preliminary study to shed some light on these, and possibly other, research questions (RQs). With such a preliminary review and analysis one can get a better focus on subsequent reviews due to a better understanding of how the field is partitioned in streams of research, who is working on what and how different topics relate to each other.

Our main assumption is that it is possible to use FCA to answer questions. FCA is a mathematical theory for data analysis that derives ontologies from sets of objects and their properties. It can be used to explore a domain, reason about it or simply describe it in a structured way. We rely on FCA visualization capabilities to help reason about the field to be surveyed.

The contributions of this paper are:

- the exemplification of how can FCA be used in preparing research literature reviews;
- answers to the RQs used to illustrate the technique.

For the illustration of the preliminary survey approach, we formulate the following RQs about SA.

1) What are the most supported definitions of software architecture?
2) What are the most popular research topics in software architecture?
3) What are the most relevant quality attributes of a software architecture?
4) What are the topics that researchers point out as being more interesting to explore in the future?

The remainder of this paper is structured as follows. Section II introduces FCA and the lattices used later on in the paper. Section III describes the proposed approach to a preliminary literature review. Each of the following four sections is an illustration of how the generic approach can be applied to a specific RQ. The paper terminates with a discussion of the outcome of the study in Section VIII.

II. Formal Concept Analysis

This section provides background on FCA in general, and the interpretation of the lattices used in the paper. Readers familiar with FCA may want to skip this section altogether.

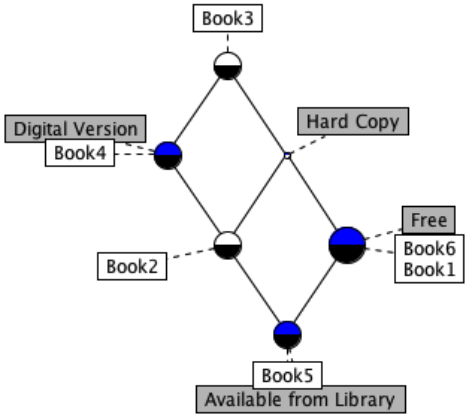


Figure 1. Example concept lattice

FCA comes from the field of applied mathematics and it is described by its proponent as follows:

> *The aim and meaning of Formal Concept Analysis as mathematical theory of concepts and concept hierarchies is to support the rational communication of humans by mathematically developing appropriate conceptual structures which can be logically activated.* [39]

The input for FCA is a *context*, which is a relation (typically a matrix) between *objects* and *attributes*. This relation forms an ontology of *concepts* and their *relationships*. A concept can be considered a *subconcept* if its *extension* (the set of its objects) is contained in the extension of its *superconcept* or, dually, if its *intension* (the set of its attributes) contains the intention of its superconcept [39].

Although FCA provides advanced features for knowledge exploration, in our analysis we only use it for visualizing *concept lattices*, which provide a structured view of the concepts under study. We are interested in understanding which objects (papers in our case) relate to which attributes (the different attributes we defined for each RQ), what clusters of objects and attributes are there, and what hierarchical relations exist among concepts. We refer the reader to a paper [41] where FCA was used for the same purpose.

To create the concept lattices in this study, we chose to use the "The Concept Explorer" (`conexp`) [1] tool. Let us start by showing how to interpret a concept lattice in FCA. Figure 1 depicts one such concept lattice where books (objects) are related to their properties (attributes). The figure shows 6 concepts (nodes in the lattice) of 3 different types. White labels represent objects, whereas gray labels represent attributes. The dashed lines between concepts and labels are a visual aid to help identify to which concepts labels belong. Solid lines represent the super/sub-concept relationship. Concepts depicted by half white, half black circles only refer to objects, meaning that objects in such nodes have empty intentions (such is the case of the top most concept where *Book3* sits) because it is not related to any of the attributes, or that the objects in that concept inherit the attributes from their superconcepts (the case of the concept where *Book2* sits, which inherits the attributes DIGITAL COPY and HARD COPY from its two superconcepts). Concepts depicted by half blue, half black circles refer to both objects and attributes (see e.g. the concept where *Book4* sits). The objects in these concepts are related to the attributes that sit on the same concept, and to the attributes of their superconcepts. Finally, the concepts depicted by a rather small and empty circle can refer to attributes (the case of the concept where the attribute HARD COPY sits), or merely serve as hierarchical links between their superconcepts and their subconcepts.

[1] http://conexp.sourceforge.net

III. GENERIC APPROACH

The approach proposed in this paper is carried out in four steps: paper selection, paper classification, use of FCA, and analysis of results. This section gives an overview of these steps.

A proper selection of papers should have strictly defined criteria for searching and filtering papers. However, because this is simply a preliminary study we skipped such a structured selection and from a set of 4 papers we chased bibliography references until we reached a larger set that both contained different types of papers (such as surveys, evaluations, new proposals, etc), and covered several different topics within the field of SA. We ended up with a selection of 38 papers published between 1992 and 2010. This set of papers is not meant to be fully representative of the field of SA, but it is vast enough for a preliminary study such as this one.

With the set of papers to survey established it is necessary to classify each paper according to the attributes selected for each of the RQs. Different sets of attributes are used to answer different RQs. For each RQ, a first (more extensive) set of attributes is manually collected from all papers. The collected attributes were found in abstract, keywords, introduction, future work and conclusion sections. Then, the related attributes are grouped together in order to reduce the number of attributes in the final set. Attributes are dropped if too few papers support them. This is typically the case for attributes that are not related to more than 1 or 2 papers whenever there are more attributes that get related to a significantly larger number of papers. Once this set is stable, then a relation is built between each paper and the attributes that are relevant to that paper.

The following step is to apply FCA. There are several tools that implement FCA. As mentioned previously, we use `conexp`, an open source FCA tool.

Finally, the resulting concept lattices are analyzed and interpreted, so as to focus on the structural relations they

uncover between papers and attributes.

The following four sections report the results of applying this generic approach to the selected RQs. The set of papers is the same for all the RQs, except for one where it was further reduced.

IV. What are the most supported definitions of software architecture?

A. Attributes

The attributes for this RQ are the following.

Design	SA is the set of design decisions, specification or otherwise abstract representation of the software system.
Structure	SA is the structure of components, their relationship and external properties.
Constraints	SA defines the constraints on the realization or satisfaction of properties of its elements, or on the legal and illegal patterns in the communication between elements.
Quality	SA influences and/or determines quality attributes of software.

B. Results

The lattice of Figure 2 shows that a group of 8 papers does not support any of the 4 given definitions of SA, as they sit in the topmost concept. Some papers are related to just one definition of SA. These are the papers that inhabit the concepts that also hold the attributes QUALITY (3 papers), DESIGN (6 papers) and STRUCTURE (11 papers). All other papers support two definitions simultaneously. Regarding attributes, the odd one out is CONSTRAINTS because no paper supports it alone. The papers that support the definition CONSTRAINTS also support DESIGN or STRUCTURE. Finally, no paper supports more than two definitions.

C. Discussion

The initial scan of the papers revealed 15 definitions. This number was, however, reduced to 4. Some definitions were simply dropped because too few papers supported them, and others were merged together. An example of such a merge is the case of STRUCTURE, which aggregates three of the initial definitions, namely STRUCTURE, DECOMPOSITION and RELATIONSHIP. The reason for this aggregation is that most papers that support these definitions articulate them together in sentences like "*... software architecture of a program or computer system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them*" *Bengtson et al. 2004* [8].

We found that 8 papers do not support any of the analyzed definitions, 5 of these not providing any definition at all. The remaining 3 papers do provide definitions, but these were dropped. Eight of the papers that provide a definition for SA quote it from elsewhere.

It turns out that STRUCTURE is the most supported (17 papers) definition among the papers we surveyed, followed by DESIGN (13 papers). Recall the definitions of DESIGN and STRUCTURE from Section IV-A. These definitions, although not exactly the same, look fairly similar to us. DESIGN refers to decisions and abstract representations. On the other hand, STRUCTURE refers to a structure of components that are related to each other in some way. We think that the decomposition of a software system in different components that are structurally related can be viewed as an abstract representation of that system. Also, such structure of components exists due to decisions made by people developing the system. This understanding of the definitions, however, does not make them the same thing. The structure of components can be observed at different abstraction levels, such as the structure of source code packages (or equivalent) which are already highly concrete and far away from the abstract representation. The point is that SA viewed from the STRUCTURE perspective can be too detailed to be considered abstract. All in all, we believe that although there might be some overlapping these are two different definitions of SA.

V. What are the most popular research topics in software architecture?

A. Attributes

The attributes for this RQ are the following.

Notation	Addresses a notation (or description language) for representing SA.
Method	Addresses a method for analysis or evaluation of SAs.
Tool	Addresses a tool for the analysis or evaluation of SAs.
Evaluation	Pertains to SA evaluation with respect to one or more quality attributes.
Analysis	Pertains to SA analysis not leading to appreciation of quality attributes.
Scenarios	Addresses scenarios as a technique for evaluation or analysis of SA.
Metrics	Addresses metrics as tools for evaluation or analysis of SA.
Reviews	Addresses reviews as tools for evaluation or analysis of SA.
Prototypes	Addresses prototypes as tools for either evaluation or analysis of SA.

B. Results

Due to the overly complex concept lattice obtained if using the entire attribute set we decided for simplification in detriment of a complete view of the concepts. To this end, we consider a core set of topics containing NOTATION, EVALUATION, ANALYSIS, TOOL and METHOD. This leaves out SCENARIOS, METRICS, REVIEWS and PROTOTYPES as

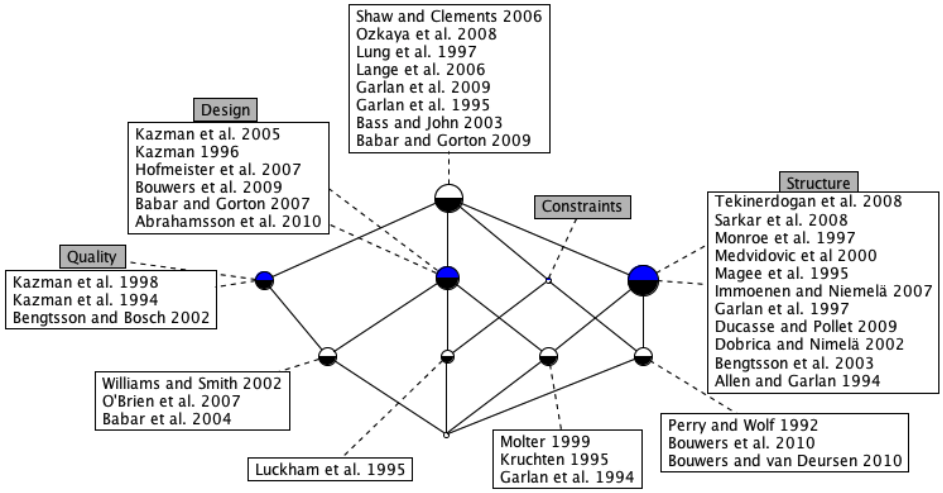


Figure 2. Concept lattice for software architecture definitions.

we expect these to be bound to some of the topics of the core set.

Figure 3 depicts the concept lattice built from classifying each surveyed paper according to the attributes of the core set. This lattice shows that there is one paper (*Shaw and Clements 2006* [35]) that does not cover any of the 6 analyzed topics. In the lattice it is visible that 5 concepts descend directly from the topmost. For easy reference we will refer to these as *level1* concepts. Four of these hold both objects and attributes, and one only holds the attribute TOOL, which means that no paper covers the TOOL topic alone. On the other hand, the 4 that hold both objects and attributes reveal that some papers are focused on just one topic. These topics are EVALUATION (9 papers), NOTATION (6 papers), METHOD (1 paper) and ANALYSIS (4 papers). Moving further down along the lattice it becomes clear that the number of connections among concepts increases. Two of the concepts that descend directly from *level1* concepts hold papers, and both have two direct superconcepts. *Kazman 1996* [20] covers TOOL and ANALYSIS, whereas a group of 10 papers cover EVALUATION and METHOD. Analyzing the lattice from the bottom up, there are 4 concepts that are superconcepts of the bottommost concept and only hold papers. All these papers cover some combination of three topics. No paper covers all topics and every topic is associated to at least one paper.

Adding METRICS to the analysis produces the concept lattice depicted in Figure 4 in which we see that METRICS are covered by 4 papers. All of these papers cover METRICS in addition to some other topic(s). For example *Bouwers et al. 2009* [10] also covers EVALUATION.

Taking SCENARIOS into account together with the core set yields the lattice of Figure 5. Nine papers cover SCENARIOS. A group of 6 papers cover it together with EVALUATION and METHOD. *Lung et al. 1997* [26] is not part of this set because it also covers NOTATION. *Kruchten 1995* [23] covers SCENARIOS together with ANALYSIS, METHOD and NOTATION. Finally, *Kazman 1996* [20] also covers TOOL and ANALYSIS.

The lattice resulting from adding REVIEWS to the analysis is depicted in Figure 6. REVIEWS are covered only by 3 papers, all also covering EVALUATION and METHOD.

PROTOTYPES, the last attribute, generates the lattice of Figure 7. Only *Luckham et al. 1995* [25] covers this attribute and it does so in addition to NOTATION.

C. Discussion

EVALUATION is the topic gathering most papers (21), followed by METHOD (17) and NOTATION (11). The difference in the number of papers that cover EVALUATION (21) and the number of papers that cover ANALYSIS (8) seems to indicate that the research community represented in the surveyed papers believes that attributing quality notions to SA is of paramount importance. On the other hand, it could also mean that there are sufficiently mature analysis techniques available and that quality attributes are the next logical step. The most frequent (12 papers) combination of two topics is EVALUATION and METHOD, which indicates that a good deal of papers focuses on methods for evaluation of SAs. Except for *Kazman 1996* [20] all papers that cover more than one topic, cover METHOD. This indicates that methods for SA evaluation or analysis are still a hot topic in the field. It could be seen as a consequence of the definition of the attribute METHOD. However the attribute TOOL was defined in a similar way and does not gather as many papers. This is perhaps a sign of method immaturity (too early for tool development).

With the exception of one paper (*Kazman 1996* [20]), all other papers that cover METRICS also cover EVALUATION. This reveals that researchers consider metrics to be adequate

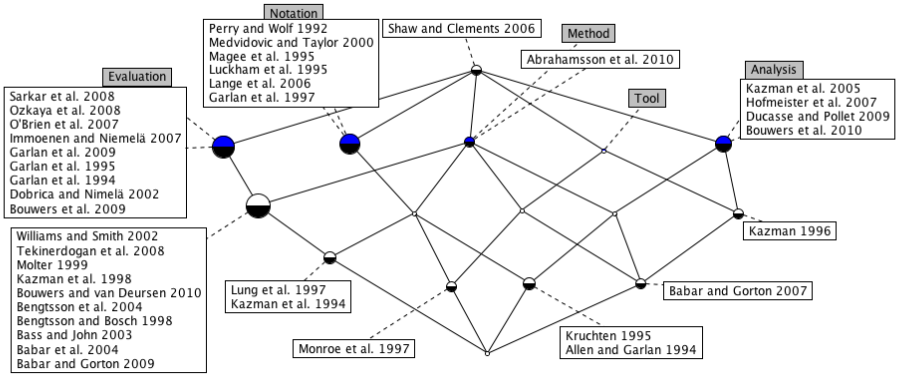


Figure 3. Concept lattice for the core set of research topics.

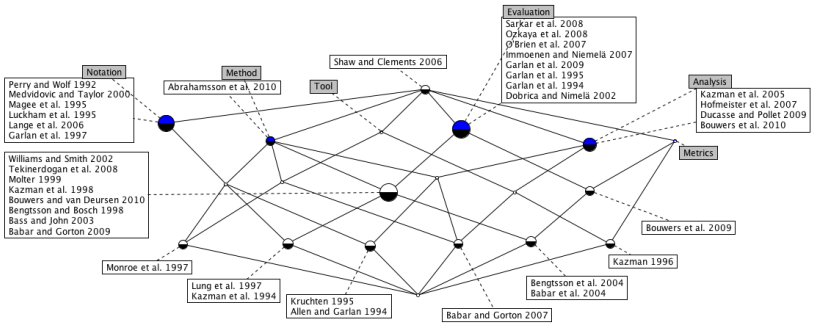


Figure 4. Concept lattice for the core set of research topics plus METRICS.

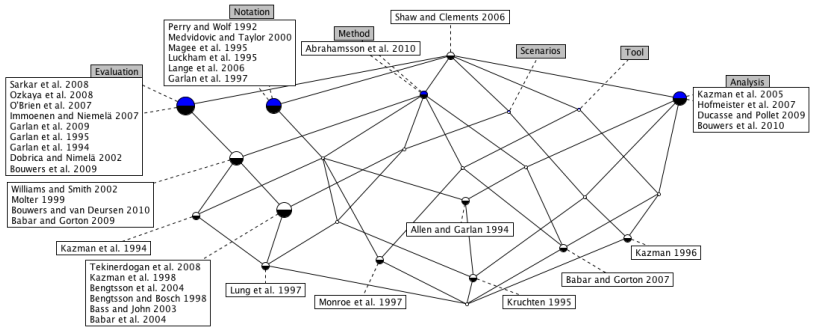


Figure 5. Concept lattice for the core set of research topics plus SCENARIOS.

indicators (or predictors) for quality attributes. Similarly to METRICS, REVIEWS are only covered together with EVALUATION and METHOD. When compared to METRICS and REVIEWS, SCENARIOS are more diversified in the field. Figure 5 shows that SCENARIOS are related to all other topics in the core set, which implies a broad scope of applications. Finally, PROTOTYPES are always covered together with NOTATION and nothing else. This does not mean researchers find prototyping inadequate for evaluation or analysis. Instead, modeling and building prototypes seems

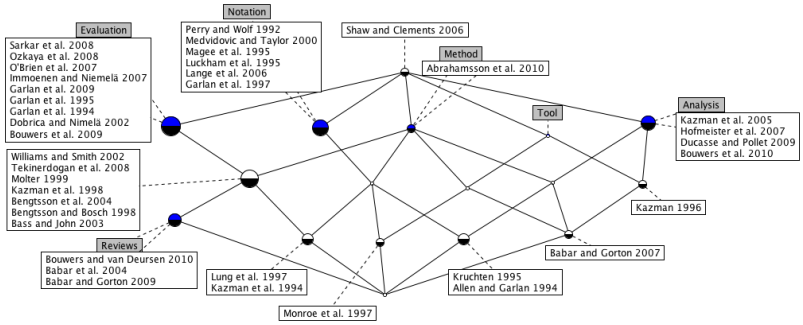


Figure 6. Concept lattice for the core set of research topics plus REVIEWS.

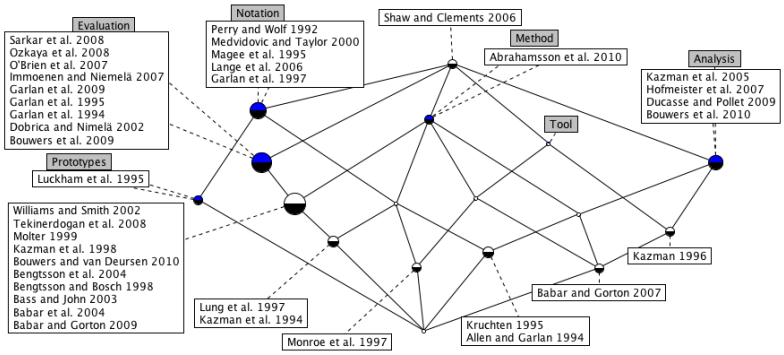
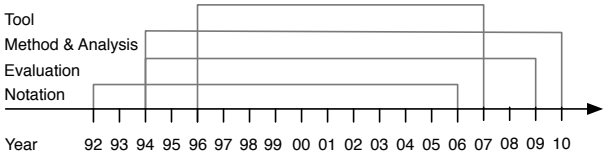


Figure 7. Concept lattice for the core set of research topics plus PROTOTYPES.

in most cases inherent to any of these activities. This could explain why researchers do not mention it explicitly when covering topics such as EVALUATION, ANALYSIS, METHOD or TOOL.



Timeline of researched topics.

From the surveyed papers we do not see a clear evolution in research topics, meaning that most topics were and are researched in parallel. As depicted in the above timeline all topics overlap along most of the timeline.

VI. WHAT ARE THE MOST RELEVANT QUALITY ATTRIBUTES OF A SOFTWARE ARCHITECTURE?

A. Attributes

The attributes for this RQ are the following.

Maintainability How easily can software be maintained.

Usability How easily can software be used.

Reusability How easily can software (or parts of it) be re-used.

Performance The amount of useful work accomplished by software compared to the time and resources used.

Reliability The capability of software to maintain performance under stated conditions for a stated period of time.

Availability The capability of a software product to be in a functioning state.

Security The capability of software to prevent unintended usage and unintended access to its data.

Modifiability How easily can software be modified.

Interoperability How easily can software interact with other systems.

Testability How easily can software be tested.

Scalability The capability of software to handle growing work loads without prejudice of its service quality.

Generic	The quality attribute is a parameter in the evaluation.

B. Results

For this RQ the set of papers was reduced to the 22 papers that cover the topic EVALUATION which, by definition of the topic, refer to some quality attribute, leading to the lattice of Figure 8. This lattice shows that all papers cover at least one quality attribute. Out of the 22 papers 18 cover at most one quality attribute (including the GENERIC). Which only leaves 4 papers that cover more than one attribute. Two papers, *Immonen and Niemelä* [19] and *Lung et al. 1997* [26], cover two quality attributes (respectively RELIABILITY and AVAILABILITY, and MODIFIABILITY and REUSABILITY). *Babar et al. 2004* [5] covers 6 quality attributes. Finally, the paper *O'Brien et al. 2007* [31] covers 9 attributes. From the perspective of the quality attributes, three (SCALABILITY, TESTABILITY, and INTEROPERABILITY) are covered by a single paper (*O'Brien et al. 2007* [31]). SECURITY is always covered together with other quality attributes (RELIABILITY, AVAILABILITY, PERFORMANCE, and USABILITY). AVAILABILITY is always covered together with RELIABILITY. The last odd attribute is GENERIC for which the associated papers do not cover any other quality attribute (this is actually a meta quality attribute as explained in Section VI-A).

C. Discussion

The fact that AVAILABILITY and RELIABILITY are always covered together hints at the similarities between these two quality attributes. In fact a software system that is very reliable should have no problems in being available to its users. However, the contrary might not be true, since a system can be available but in the end not so reliable. For instance a system could be available for its users to perform whatever tasks they need it for, but still lose important data without the user perceiving it. This seems to indicate AVAILABILITY as an aspect of RELIABILITY.

Because SECURITY is always covered together with RELIABILITY, AVAILABILITY, PERFORMANCE and USABILITY it does not seem a main concern of researchers focusing on SA. The fact that it is covered by no more than two papers seems to reinforce this interpretation.

Seven papers cover GENERIC evaluations, meaning that the quality attribute is a parameter of the evaluation method, therefore these methods can be applied to different quality attributes. The next most covered quality attribute is REUSABILITY (6 papers) which indicates this quality attribute as the most relevant for the surveyed papers. This makes sense if one considers that most SAs are built from scratch every time, over and over. Just like with code libraries, researchers believe that there should be ways to reutilize existing SAs, or at least bits and pieces. Lastly, we have MAINTAINABILITY and RELIABILITY each with 4.

The remaining quality attributes are covered by three or two papers, with the exception of INTEROPERABILITY and TESTABILITY. A possible explanation for these two quality attributes to appear in isolation from the others, could be the development stage of the SA they apply to. There is a distinction between *designed* and *implemented* architecture. The former refers to the architecture that was initially designed for a system typically using an Architecture Description Language (ADL), whereas the latter refers to the architecture that got implemented in source code. It could be that both INTEROPERABILITY and TESTABILITY can be better assessed when considering implemented architectures.

VII. WHAT ARE THE TOPICS THAT RESEARCHERS POINT OUT AS BEING MORE INTERESTING TO EXPLORE IN THE FUTURE?

A. Attributes

The attributes for this RQ are the following.

Notation	Improve notation support for SA, including enrichment of semantics.
Knowledge	Create and/or extend a reusable SA body of knowledge.
Validation	Validate usefulness of methods, tools and languages used for evaluation and analysis of SAs.
Evaluation	Improve SA evaluation methods.
Tooling	Improve SA tool support.
Quality	Study SA quality attributes.
Integration	Promote integration of SA evaluation and analysis methods in software development processes.
Measurement	Propose new, or select from existing, metrics for SA.

B. Results

The lattice of Figure 9 shows that all topics sit in concepts that directly descend from the topmost concept. This means that the topics are fairly independent. A group of 6 papers sits in the topmost concept, meaning that they either do not propose any future work, or their topics cannot be found in the attribute set. No paper proposes EVALUATION or TOOLING alone. All other topics are singled out in some papers. Before moving on to the remaining 20 papers that propose several topics, one noteworthy observation is that VALIDATION is the topic which is most proposed together with other topics (11 papers), followed by INTEGRATION (10 papers), and TOOLING (7 papers). The 17 papers that propose multiple topics do so in pairs.

C. Discussion

When compiling the attribute set for this RQ, we encountered two papers (*Perry and Wolf 1992* [33] and *Babar et al. 2004* [5]) which proposed a working definition of

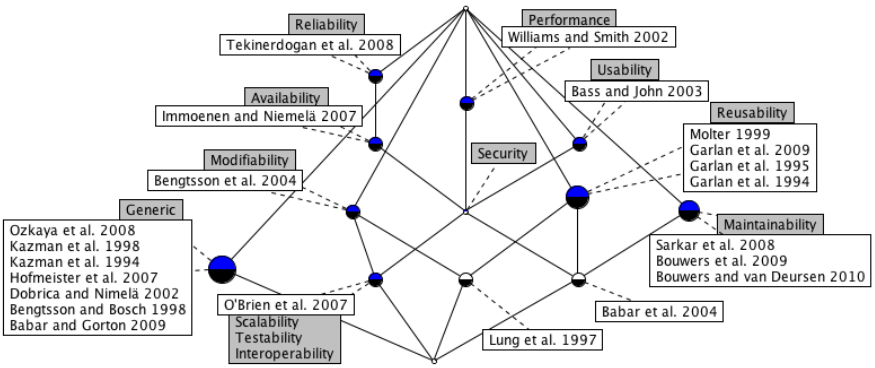


Figure 8. Concept lattice for the most explored quality attributes.

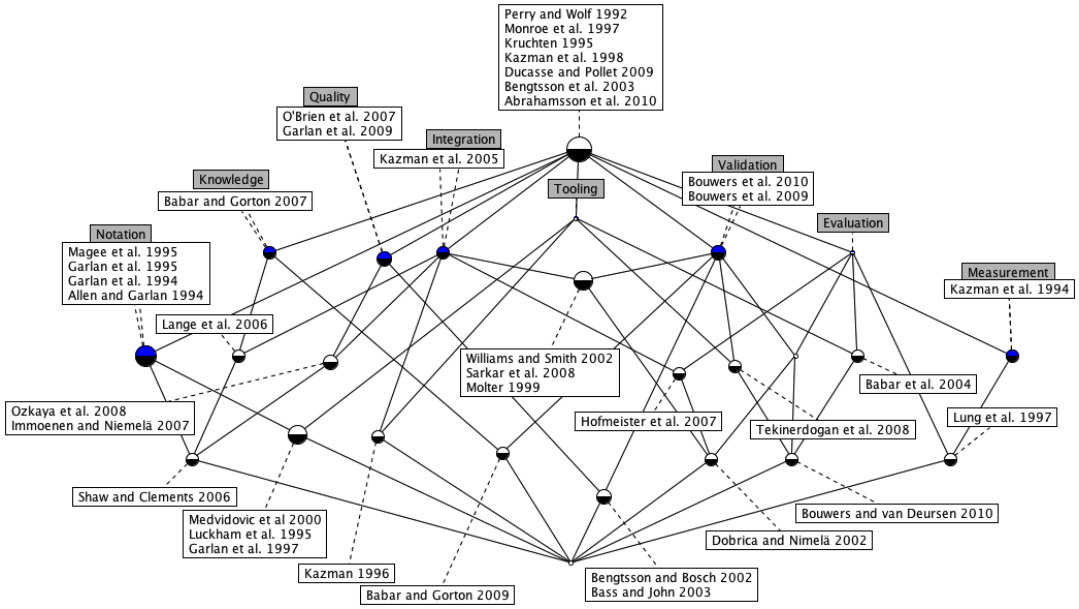


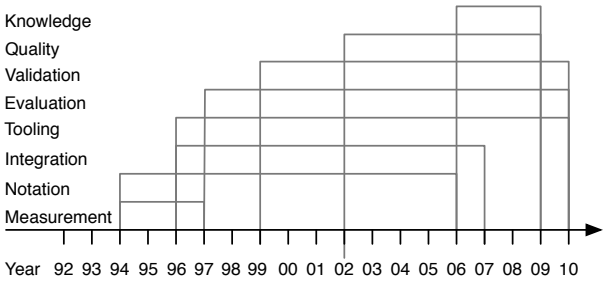
Figure 9. Concept lattice for the research topics that researchers point out as more interesting to explore in the future.

SA. Although this topic did not make it into the analyzed attribute set, it shows the relevance of our first RQ.

The pairs of topics that were pointed out by more papers (3 each) were NOTATION and TOOLING, and INTEGRATION and VALIDATION. This seems to stress the need for tool support for SA specific notations, and that methods and tools need to be both validated with respect to their usefulness and better integrated in development processes.

Six papers do not propose future work topics at all. The most proposed topics VALIDATION and INTEGRATION (11 papers each), by definition, only make sense when researched together with something else. Even though some papers single these two out, we believe that the intention of the authors is to validate or integrate whatever methods or tools they have developed. All in all, it appears that there are many avenues to be explored by new research and that there is no topic that gathers the majority of the researchers.

Contrary to what we found for RQ2, there is not as much time overlapping between proposals for future research topics. The following timeline shows that both MEASUREMENT and KNOWLEDGE span over short periods of time 1994–1997 and 2006–2009, respectively. For the remaining topics there is significant overlap for most of the analyzed period, which means that for this RQ there is no crystal clear evolution in what researchers consider to be important to research in the future.



Timeline of future research topics.

VIII. Conclusions

Our preliminary survey shows that the term SA lacks a clear definition. Two of the surveyed papers actually mention the desirable clarification of the term as future work. From the analysis of RQ1 ("*What are the most supported definitions of software architecture?*") we were able to single out 2 (DESIGN and STRUCTURE) of the 4 main definitions as being the most supported. This dichotomy can be explained by the gap existing between the designed architecture (what is initially designed for a system) and the implemented architecture (what is actually implemented in the source code). In a full fledge survey of the field it might be desirable to take both definitions into account, or to just focus on one. If one already knows the field to be surveyed, such differences might be obvious. However, if the field is unknown, a preliminary study as the one described in this paper can help to focus on what to include and what to exclude from the survey.

With respect to RQ2 ("*What are the most popular research topics in software architecture?*"), one thing that stands out is the fact that whenever a paper covers two topics, METHOD is almost always (there is one exception) one of them. So if the focus of the survey would be "methods for SA", one would already expect not to exclude any topic from the literature selection. On the other hand, should the focus of the survey be "tools for dealing with SA", one would already expect to find more papers regarding methods for architectural analysis that than, for instance, architectural evaluation. Of course, these are not rules of thumb, but they help set the expectations, which in turn guide literature searches and help validate findings. RQ2 also shows how to partition the analysis if the lattice is overly complex.

Regarding RQ3 ("*What are the most relevant quality attributes of a software architecture?*"), we observed that the maintainability, reusability, usability, performance, reliability and modifiability quality attributes seem to be more impacted by decisions taken at the level of SA. Furthermore, other quality attributes, such as scalability, testability and interoperability, seem to be less relevant according to the number of papers that cover those.

Finally, the analysis of RQ4 ("*What are the topics that researchers point out as being more interesting to explore in the future?*") shows that there are many avenues for future work and that there are many proposals for combining these avenues into streams of research. From all the identified future research topics the ones that gather more consensus are validation and integration efforts. This seems to indicate that the field has matured to the point where there is enough confidence in what has been developed thus fur to start embracing integration of such methods and tools into mainstream software engineering practices. On the other hand, the field has not yet matured enough to have provided full confidence in its methods and tools, thus requiring additional empirical validation studies.

We recognize limitations in this preliminary study for a survey, namely the bias introduced in the selection of papers and attributes. To overcome this bias, one should rely on strictly defined criteria for searching and selecting papers, and the elicitation of the attributes. Since the goal of this study is the demonstration of how FCA can help in preparing for a full fledge survey, we do not consider this bias to be a problem. The way to structure such criteria and protocols for searching, classifying and extracting knowledge from scientific literature has already been addressed in systematic literature reviews for software engineering [40].

From this study we conclude that FCA can help in gathering knowledge about a multifaceted field, and better focus a survey. In addition, we believe that this is also true if the target of the survey is something more specific, such as "methods for SA evaluation that use metrics". Validating this hypothesis is yet to be done.

References

[1] P. Abrahamsson, M. Babar, and P. Kruchten, "Agility and architecture: Can they coexist?" *IEEE Softw.*, vol. 27, no. 2, pp. 16 –22, Mar/Apr 2010.

[2] R. Allen and D. Garlan, "Formalizing architectural connection," in *ICSE*, May 1994, pp. 71 –80.

[3] M. Babar and I. Gorton, "A tool for managing software architecture knowledge," in *SHARK/ADI*. IEEE, 2007, p. 11.

[4] M. A. Babar and I. Gorton, "Software architecture review: The state of practice," *IEEE Comp.*, vol. 42, no. 7, pp. 26–32, 2009.

[5] M. A. Babar, L. Zhu, and R. Jeffery, "A framework for classifying and comparing software architecture evaluation methods," in *ASEC*, ser. ASWEC '04. IEEE CS, 2004, pp. 309–.

[6] L. Bass and B. E. John, "Linking usability to software architecture patterns through general scenarios," *JSS*, vol. 66, no. 3, pp. 187 – 197, 2003.

[7] P. Bengtsson and J. Bosch, "Scenario-based software architecture reengineering," in *ICSR*. IEEE, 2002, pp. 308–317.

[8] P. Bengtsson, N. Lassing, J. Bosch, and H. van Vliet, "Architecture-level modifiability analysis (ALMA)," *JSS*, vol. 69, no. 1-2, pp. 129 – 147, 2004.

[9] E. Bouwers and A. van Deursen, "A lightweight sanity check for implemented architectures," *IEEE Softw.*, vol. 27, no. 4, pp. 44 –50, Jul 2010.

[10] E. Bouwers, J. Visser, and A. van Deursen, "Criteria for the evaluation of implemented architectures," in *ICSM*. IEEE, 2009, pp. 73–82.

[11] E. Bouwers, J. Visser, C. Lilienthal, and A. van Deursen, "A cognitive model for software architecture complexity," in *ICPC*. IEEE Comp. Soc., 2010, pp. 152–155.

[12] L. Dobrica and E. Niemelä, "A survey on software architecture analysis methods," *IEEE TSE*, vol. 28, pp. 638–653, Jul 2002.

[13] S. Ducasse and D. Pollet, "Software architecture reconstruction: A process-oriented taxonomy," *IEEE TSE*, vol. 35, no. 4, pp. 573–591, 2009.

[14] D. Garlan, R. Allen, and J. Ockerbloom, "Architectural mismatch: Why reuse is so hard," *IEEE Softw.*, vol. 12, pp. 17–26, Nov 1995.

[15] ——, "Architectural mismatch or why it's hard to build systems out of existing parts," in *ICSE*, 1995, pp. 179–185.

[16] D. Garlan, R. Monroe, and D. Wile, "Acme: an architecture description interchange language," in *CASCON*, ser. CASCON '97. IBM Press, 1997, pp. 7–.

[17] D. Garlan, R. Allen, and J. Ockerbloom, "Architectural mismatch: Why reuse is still so hard," *IEEE Softw.*, vol. 26, no. 4, pp. 66 –69, Jul/Aug 2009.

[18] C. Hofmeister, P. Kruchten, R. L. Nord, H. Obbink, A. Ran, and P. America, "A general model of software architecture design derived from five industrial approaches," *JSS*, vol. 80, no. 1, pp. 106 – 126, 2007.

[19] A. Immonen and E. Niemelä, "Survey of reliability and availability prediction methods from the viewpoint of software architecture," *SSM*, vol. 7, pp. 49–65, 2008.

[20] R. Kazman, "Tool support for architecture analysis and design," in *ISAW-2 and Viewpoints '96 on SIGSOFT '96 Workshops*, ser. ISAW '96. ACM, 1996, pp. 94–97.

[21] R. Kazman, L. Bass, M. Webb, and G. Abowd, "SAAM: A method for analyzing the properties of software architectures," in *ICSE*. IEEE CS, 1994, pp. 81–90.

[22] R. Kazman, L. Bass, M. Klein, T. Lattanze, and L. Northrop, "A basis for analyzing software architecture analysis methods," *SQJ*, vol. 13, pp. 329–355, 2005.

[23] P. Kruchten, "The 4+1 view model of architecture," *IEEE Softw.*, vol. 12, no. 6, pp. 42 –50, Nov 1995.

[24] C. Lange, M. Chaudron, and J. Muskens, "In practice: UML software architecture and design description," *IEEE Softw.*, vol. 23, no. 2, pp. 40 – 46, Mar/Apr 2006.

[25] D. Luckham, J. Kenney, L. Augustin, J. Vera, D. Bryan, and W. Mann, "Specification and analysis of system architecture using rapide," *IEEE TSE*, vol. 21, no. 4, pp. 336 –354, Apr 1995.

[26] C.-H. Lung, S. Bot, K. Kalaichelvan, and R. Kazman, "An approach to software architecture analysis for evolution and reusability," in *CCASCR*, ser. CASCON '97. IBM Press, 1997, pp. 15–.

[27] J. Magee, N. Dulay, S. Eisenbach, and J. Kramer, "Specifying distributed software architectures," in *ESEC*, ser. LNCS, W. Schäfer and P. Botella, Eds. Springer Berlin / Heidelberg, 1995, vol. 989, pp. 137–153.

[28] N. Medvidovic and R. Taylor, "A classification and comparison framework for software architecture description languages," *IEEE TSE*, vol. 26, no. 1, pp. 70 –93, Jan 2000.

[29] G. Molter, "Integrating SAAM in domain-centric and reuse-based development processes," in *NOSA*, 1999, pp. 1–10.

[30] R. Monroe, A. Kompanek, R. Melton, and D. Garlan, "Architectural styles, design patterns, and objects," *IEEE Softw.*, vol. 14, no. 1, pp. 43 –52, Jan/Feb 1997.

[31] L. O'Brien Lero, P. Merson, and L. Bass, "Quality attributes for service-oriented architectures," in *SDSOA*, May 2007, pp. 3–3.

[32] I. Ozkaya, L. Bass, R. Nord, and R. Sangwan, "Making practical use of quality attribute information," *IEEE Softw.*, vol. 25, no. 2, pp. 25–33, 2008.

[33] D. E. Perry and A. L. Wolf, "Foundations for the study of software architecture," *SIGSOFT SEN*, vol. 17, pp. 40–52, Oct 1992.

[34] S. Sarkar, A. C. Kak, and G. M. Rama, "Metrics for measuring the quality of modularization of large-scale object-oriented software," *IEEE TSE*, vol. 34, no. 5, pp. 700–720, 2008.

[35] M. Shaw and P. Clements, "The golden age of software architecture," *IEEE Softw.*, vol. 23, no. 2, pp. 31 – 39, Mar/Apr 2006.

[36] B. Tekinerdogan, H. Sozer, and M. Aksit, "Software architecture reliability analysis using failure scenarios," *JSS*, vol. 81, no. 4, pp. 558 – 575, 2008.

[37] L. G. Williams and C. U. Smith, "PASASM: a method for the performance assessment of software architectures," in *IWSP*, ser. WOSP '02. ACM, 2002, pp. 179–189.

[38] P. Clements, R. Kazman, and M. Klein, *Evaluating Software Architectures: Methods and Case Studies*. Addison-Wesley, 2001.

[39] R. Wille, "Formal concept analysis as mathematical theory of concepts and concept hierarchies," in *FCA*, ser. LNCS, B. Ganter, G. Stumme, and R. Wille, Eds., vol. 3626. Springer, 2005, pp. 1–33.

[40] B. Kitchenham, "Evidence-based software engineering and systematic literature reviews," in *PROFES*, ser. Lecture Notes in Computer Science, J. Münch and M. Vierimaa, Eds., vol. 4034. Springer, 2006, p. 3.

[41] J. Poelmans, P. Elzinga, S. Viaene, and G. Dedene, "Formal concept analysis in knowledge discovery: A survey," in *ICCS*, ser. LNCS, M. Croitoru, S. Ferré, and D. Lukose, Eds., vol. 6208. Springer, 2010, pp. 139–153.