Towards Software Assets Origin Selection Supported by a Knowledge Repository

Antonio Cicchetti*, Markus Borg†, Séverine Sentilles*, Krzysztof Wnuk‡, Jan Carlson* and Efi Papatheocharous†
*Mälardalen University,
Västerås, Sweden
Email: name.surname@mdh.se
†SICS Swedish ICT AB,
Kista, Sweden
Email: name.surname@sics.se
‡Blekinge Institute of Technology,
Karlskrona, Sweden
Email: krzysztof.wnuk@bth.se

***Abstract*—Software architecture is no more a mere system specification as resulting from the design phase, but it includes the process by which its specification was carried out. In this respect, design decisions in component-based software engineering play an important role: they are used to enhance the quality of the system, keep the current market level, keep partnership relationships, reduce costs, and so forth. For non trivial systems, a recurring situation is the selection of an asset origin, that is if going for in-house, outsourcing, open-source, or COTS, when in the need of a certain missing functionality. Usually, the decision making process follows a case-by-case approach, in which historical information is largely neglected. This solution avoids the overhead of keeping detailed documentation about past decisions, but hampers consistency among multiple, possibly related, decisions.**

The ORION project aims at developing a decision support framework in which historical decision information plays a pivotal role: it is used to analyse current decision scenarios, take well-founded decisions, and store the collected data for future exploitation. In this paper, we outline the potentials of such a knowledge repository, including the information it is intended to be stored in it, and when and how to retrieve it within a decision case.

I. Introduction

Software architecture is not only the final product of the design efforts, but more and more is including the decision history that brought to the system as it is finally specified [1]. One of the intended goals underlying this trend is, together with designing the system architecture, preserving domain-experts' knowledge/reasoning, which is implicitly exploited to reach the design solution and would be lost if not recorded appropriately. However, in software architecture knowledge management is a challenge [2], and recording the information about a certain decision making process is an open research question: on the one hand, the stored decision information shall be detailed enough to turn out as useful for future decision making scenarios, even in cross-application and cross-domain situations; on the other hand, the efforts for recording first, and for retrieving/analysing/maintaining past decision data later, shall be reduced to the minimum [3].

Component-Based Software Engineering (CBSE) has been recognised as an effective methodology to tackle development complexity: it is based on the reuse of units of computation called, indeed, components, appropriately assembled to build-up the final system [4]. Lately, the concept of component has been extended to include services, where the computational unit can be plugged to the system on demand [5]. Therefore, in this paper we refer more generically to *software assets* (or simply assets) as the reusable units of computation that are used to build-up a system. In this context, choosing between assets origins is an architectural aspect that can have important side effects in the future history of the system. In particular, assets origins can be distinguished in: i) in-house development, ii) outsourcing, iii) open source, iv) COTS. In general, the choice of a particular origin may not be straightforward; on the contrary, it might be affected by a number of functional and extra-functional requirements related to the system. Even more intricate, in several decision scenarios contextual factors (e.g. partnerships) might come into play, making an apparent solution not optimal. In this respect, it is of paramount important to carefully analyse the available choices and their corresponding implications. Moreover, the rationale underlying the final decision should be recorded both to keep a certain degree of consistency among related decisions and to trace back decisions outcomes.

The ORION project [6] aims at providing support in the decision making process, when opting between one of the four asset origins mentioned above, by coordinating the relevant information that should be taken into account in order to take a well-founded decision. In other words, the ORION solution does not aim at automatically deriving the most suitable origin alternative, rather it is meant to aid the necessary activities targeting the collection of enough *evidence* for justifying a certain asset origin choice. In this context, we believe that the existence of a knowledge repository is a necessary condition. In fact, it allows to query previous decision experience and therefore explore what aspects were considered as relevant for past choices.

In this paper we discuss how a knowledge repository can be used in a decision process as the one proposed by ORION. In particular, ORION is based on several phases in the decision making process, and for each phase we illustrate what kind of historical knowledge might be relevant. Moreover, we investigate the necessary trade-offs between amount of stored decision information and corresponding ways of exploiting the available information for supporting current decisions [7]. Eventually, we also discuss some practical issues emerging from the use of our decision support in cross-application and even cross-domain scenarios.

The structure of the work is as follows: next section introduces the basic concepts related to the decision process proposed by ORION, together with related research in architectural knowledge representation. Subsequently, Section III introduces the intended contribution of this paper, that is, proposing a decision support for software assets origin selection based on a knowledge repository. In turn, the repository exploits a well-defined vocabulary to define all the necessary details about a certain decision scenario, presented in a corresponding ontology in Section IV. Section V investigates a number of technical issues that should be taken into account when dealing with a decision support systems for software engineering, knowledge representation, storage, analysis, and maintenance. Eventually, the paper culminates with conclusions about the investigations presented in this work and describes the plans for the upcoming research directions.

II. Background

A. The ORION Project

The ORION project envisions a decision process as the one shown in Figure 1. Each decision is initiated by one (or more) of its stakeholders, who establish the set of suitable asset origins together with one or more goals. A goal is a generic, possibly long-termed, objective a company is willing to achieve: therefore, goals have to be "instantiated" into corresponding selection criteria and priorities to be assigned to the goodness of the properties taken into account[1]. Once this basic information has been set-up, it is necessary to describe the context in which the decision is going to be taken. In particular, details about the company, its application domain, competitors, and so forth, need to be carefully documented. This latest activity will help in understanding the rationale behind decisions both inside and outside the company: it is not a coincidence that the context is exploited as pivotal information to retrieve past experience and feed back the decision process.

Notably, based on similar cases it is possible to check that all the relevant stakeholders have been involved, that appropriate origins have been taken into account, and that selection criteria and priorities are consistent with the initial decision goals as well as with the properties that will be evaluated accordingly. Eventually, property values and their weights are given as input to a specific decision method which will output a decision. It is worth noting that the decision method has not to be necessarily automatic, rather it could rely on expert opinions. In any case, once the final decision is taken, the whole collection of data related to the current decision process is saved in the knowledge repository.

[1]In this respect, a goal might embed requirements, architectural choices lifecycle aspects, and so forth.

B. Related works

The *why* behind design decisions culminating in a software architecture has been gaining more and more importance, even over the architecture specification itself [1]. This has caused a growing need of storing architectural knowledge in an effective way, i.e. finding a good balance between effort required in storing and maintaining architectural details and level of support in future architecting activities [7], [8].

A relevant corpus of literature has been devoted to investigating solutions for storing architectural knowledge, and a review of the approached is beyond the scope of this work. The interested reader is referred to [2], [9], [10] for detailed surveys on the subject. For the purpose of this paper, it is important to remark that (semi-)automated knowledge reasoning is still scarcely supported, because it typically requires a preliminary encoding of knowledge [2]. Such a preliminary encoding is done through the ontology discussed in Section IV. A similar approach is adopted in [11], where the authors address the management of risk knowledge related to technology sustainability, and create a corresponding ontology for encoding sustainability risks.

Tang *et al.* [7] discuss the issues raising in defining an ontology for software architecture documentation and propose a lightweight solution. In this respect, the ontology proposed in this paper is quite extensive due to the need of creating a common comparison platform for decision knowledge possibly cross-application and cross-domain. Nevertheless, it is partitioned by six key entities and related sub-categories, by which the knowledge storage can be made smoother.

Historical information can provide valuable assistance during the decision making process. Therefore, it should be structured in a form of knowledge representation containing as rich description of past cases as possible. For decision makers, the main benefits of a well-structured knowledge representation are 1) avoiding making the same suboptimal decisions, 2) highlighting rationales for making past choices, and 3) enabling comparisons across decision scenarios. The expected benefits of tailored knowledge representations are based on the hypothesis that many architectural decisions are reoccurring and therefore, if properly stored, can form valuable lessons learned. Moreover, aggregating the history of several decisions based on a given criteria may reveal interesting insights about strategic decision making directives. Software architectural knowledge, i.e. architectural design and architectural design decisions, need to be stored and managed, otherwise it becomes tacit and erodes with personnel turnover [12]. Finally, knowledge is useful when making compromises and decisions under uncertainty [13].

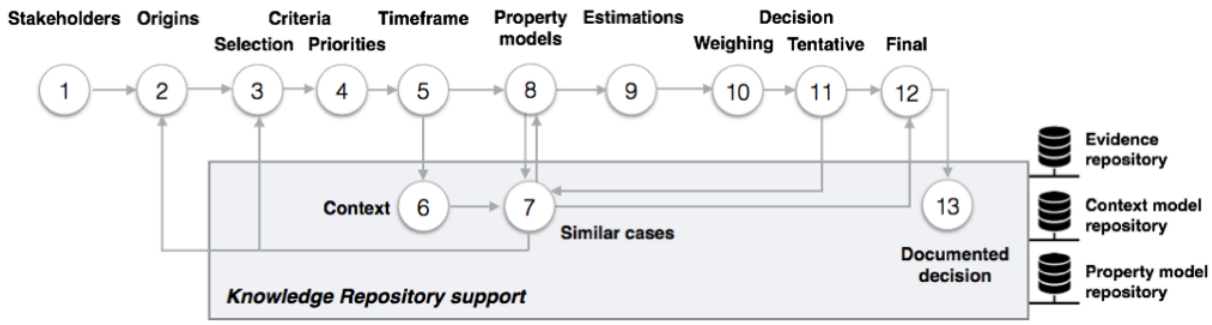

Fig. 1. The ORION decision making process [6].

Knowledge representations regarding architectural decisions have evolved over the years from "boxes and arrows" to the international standard ISO/IEC/IEEE 42010 [14], and to comprehensive architectural methods, e.g. RUP and capturing design rationales [15]. Kruchten divides architectural decisions into: 1) ontocrises - existence decisions about what structure and behavior a system should have, 2) anticrises - what elements should **not** be included in a system, 3) diacrises - what should the quality of the system be and 4) pericrises - executive decisions about the business environment, processes, tools etc. Moreover, Kruchten listed the following attributes of architectural design decisions: 1) epitome (the decision itself), 2) rationale ("why" or justification), 3) scope (by default the decisions are universal but there could also be limited scope decisions) 4) state (idea, decided, approved), 5) author, timestamp and history, 6) categories, 7) cost, 8) risk, 9) related decisions and 10) relationships with external artifacts (including traces from and traces to). The "related decision" attribute is particularly relevant for building knowledge representations and is decomposed into the following types of relationships: 1) constrains, 2) forbids, 3) enables, 4) subsumes, 5) conflicts, 6) overrides, 7) comprises, 8) is bound to, 9) is an alternative to, 10) is related to.

Tang and van Vliet listed the following types of design concerns: 1) purposes and goals, 2) functional requirements, 3) non-functional requirements, 4) business environment, 5) information system environment, 6) technology and IT environments and 7) design (the chosen design has some influence on the rest of the architecture) [16]. Design concerns 1), 2), 3) and 4) are related to Kruchten's attributes of architectural design decisions. Tyree and Akerman suggested a template for architecture decisions that includes: issue, decision, status, group, assumptions, constraints, positions, argument, implications, related decisions, related requirements, related artifacts, related principles and notes [17].

Ali Babar *et al.* introduced a model of organizing architectural knowledge that includes architecturally significant requirements associated with patterns, scenarios and architecture. This model also emphasizes architecture description as means to capturing architecture design decisions and focuses on capturing decision rationale [12]. Capilla *et al.* suggested a meta-model of architectural design decisions that also include functional and non-functional requirements supported by architecture. They recognize a decision model that involves queries, constraints and dependencies but do not provide further details. Decisions in their model have style, patterns and variation points and can be taken in iterations [18].

van Vliet *et al.* present a model of architectural knowledge that focuses around architectural design decisions related to artifacts, stakeholders, activities and concerns [19]. No patterns, requirements or associations to previous decisions or other decisions are mentioned in this model.

Falessi *et al.* [20] presented a characterization schema for decision-making techniques that divide them into problem space (that describe quality attributes requested by various stakeholders and their importance) and solution space (that describe fulfilment that each alternative offers and uncertainty that it has). This schema is enriched by a ranking table and applied to fifteen decision making techniques for COST selection. Finally, they emphasize the lack of comprehensive description of quality attributes for decision-making techniques and mapped the identified challenges onto the characterization schema.

Ontologies were used for representing decision making by several authors, i.e. data models that represent concepts in decision making as well as their relations. For example, Kornyshova and Deneckere introduced a decision-making ontology for information system engineering [21]. Their ontology includes decisions grouped in decision making situations and described by criteria. Decision consequences, alternatives, and stakeholders are also included. While there is indirect value in the learning process involved in creating an ontology, Schneider stresses that benefits are realized first when an ontology is actually used [13, pp. 127].

III. Contribution: Towards a taxonomy-supported knowledge repository

For the purpose of this work, it is important to notice that each phase of the decision process can benefit from inputs/feedbacks coming from past knowledge. In particular, the knowledge repository can serve as support for defining a detailed context (number 6 in Figure 1), to retrieve similar cases and hence explore past experiences (number 7) to refine the set of candidate component origins and the selection

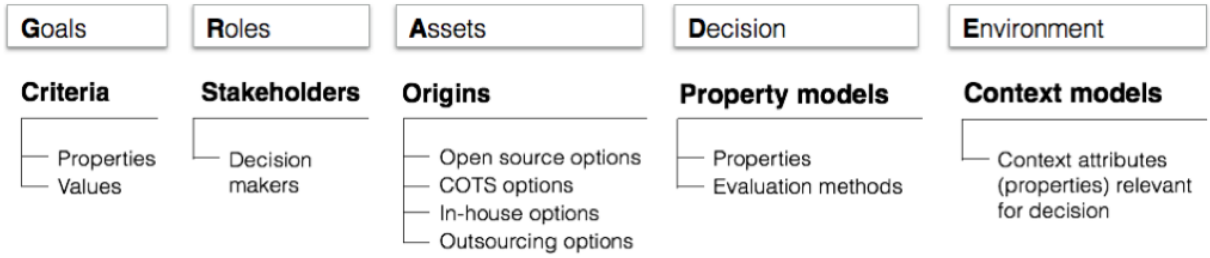

Fig. 2. The GRADE taxonomy.

itself (2 and 3 respectively), for getting feedbacks on both the tentative and final selections (11 and 12, respectively), and to eventually store the final selection decision. It is also worth noting that ORION does not prescribe any automated component selection mechanism: in fact, the knowledge is meant to be exploited to take a more informed decision, and to carefully document the process that brought to such outcome, in full compliance with the more recent notion of software architecture [1]. Even if the importance of historical data is clear, it would be of no use without a common understanding about decision information. In particular, without a consistent vocabulary across different decisions, past cases would be difficult (and even misleading) to analyse and exploit in an effective way both internally and externally to the company. In this respect, it is necessary to define a common taxonomy by which all the decision elements can be carefully taken into account and described. Therefore, hand-in-hand with ORION we proposed the GRADE taxonomy [22], which indeed introduces a common base of reasoning with respect to each particular decision. Notably, as shown in Figure 2, each decision can be characterised by multiple aspects pertaining to the overall goal of a certain decision, the roles involved in the decision making process, the properties of the involved assets, the adopted decision method, and information about the context in which the decision is taken.

Again, the role played by the knowledge repository is to support a more conscious decision process:

- Goals are mapped towards corresponding evaluation criteria. In this respect, the past knowledge can serve to extend/refine the set of criteria derived from a certain goal, and hence to adopt suitable acceptance criteria;
- Roles refer to a decision stakeholders: in every scenario, it is important that all the relevant stakeholders are considered in the decision process, in terms of both inputs and outputs of a certain decision. Therefore, the knowledge repository can help in keeping track of whom should be involved in the decision process;
- Assets entail a number of implicit and explicit consequences related, for instance, to the desired functional and extra-functional properties for the system under development and the application domain in which the selection has to be made. Moreover, historical evaluations might be useful in future decisions involving the same asset origins;
- Decision refers to the collection of relevant criteria (properties) and the evaluation of the most convenient alternative. Properties are characterised by corresponding estimation and/or evaluation methods. Moreover, the selected decision method might be to exploit experience of the decision makers, to adopt some mathematical optimisation approach, and so on. The knowledge repository can support both the selection of appropriate estimation/evaluation methods for property measurements, and the choice of a suitable decision method to be used to take the final decision;
- Environment groups the set of "background" information, called context in ORION, which can play an important role in selecting the feasible origin alternatives. Context information becomes extremely relevant when trying to understand the rationale behind a certain decision, as it can affect the criteria derived from the decision goals, the stakeholders, the feasible origin alternatives, and the selected properties and evaluation methods as well. As a consequence, a knowledge repository can improve the comprehension of the decision scenario and the effects of it on the decision process.

In the following section, it is illustrated a deeper investigation about a possible taxonomy, corresponding to the concepts defined in GRADE, appropriate to adequately represent the necessary knowledge about a certain decision in software assets origin selection.

IV. Architectural Knowledge Representation

Figure 3 shows the initial version of the ontology intended for use in the decision support system under development in the ORION project, in the form of an ER model. The ontology extends the GRADE taxonomy [22] by refining entities and introducing their inter-relations. The ontology revolves around the *Decision* entity, the centerpiece of the decision process under study in the ORION project. The Decision can be intended to be used privately or publicly (e.g., across organizational boundaries/limits). The Decision is composed of six key entities on the first level, from left to right: *Environment Aspect*, *Value Perspective*, *Asset Usage*, *Role Level*, *Decision Level*, and *Method Family*. The full details of the ontology is being prepared in a separate publication, but we continue by providing an overview description.

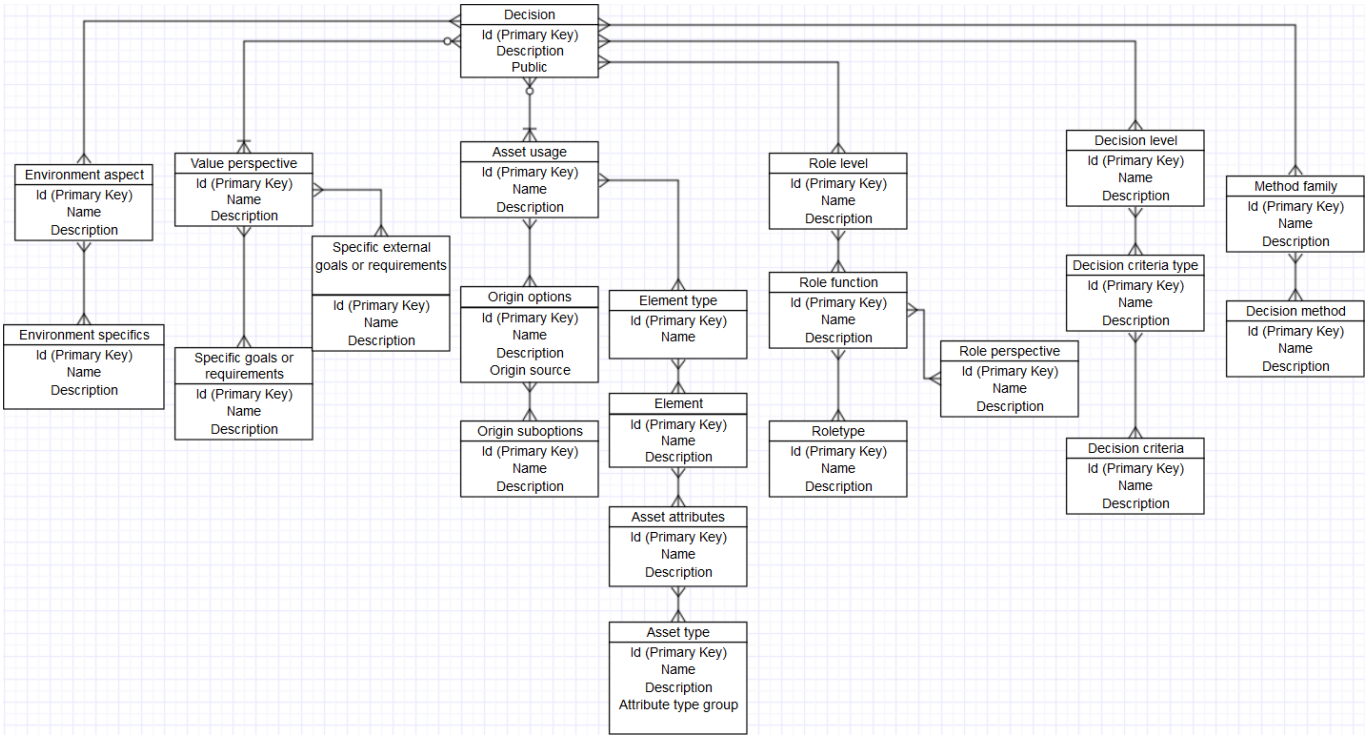

Fig. 3. The ontology proposed for the decision support system under development, presented as an ER diagram. The cardinality of the relationships is presented using the "Information Engineering" style [23].

The Environment Aspect, composed of a number of *Environment Specifics*, describes the contextual situational information in which the decision process is ongoing. It includes for example organization, market and business information. Properly describing the context of the phenomenon under study is critical in software engineering [24], and fundamental to providing actionable decision support. For our particular focus of selecting asset origins, we have so far identified 80 specifics that can be used to characterize the context, organized into five separate dimensions: organization, product, stakeholders, development methodologies, and market & business.

The Value Perspective represents the goal an organization (or individual as decision maker) wants to achieve in relation to the decision process, and includes also the requirements involved in the decision. Further distinction between the internal and external perspectives is carried out by separating the two entities *Specific Goals or Requirements* and *External Goals or Requirements*. Significant input to how the Value Perspective of the goals are categorized is the *software value map*, value perspectives identified for software-intensive product development by Khorum *et al.* [25]. A total of 34 unique goals are identified that can be targeted by a development organization, e.g. extending functionality of an existing product, improving customer satisfaction, and increasing modularity.

Asset Usage and its related entities form one of the most complex group of entities in the proposed ontology, consisting of *Origin Options* (that have related *Origin Suboptions*), *Element Type* (related with *Element*), and *Asset Type* (related with *Asset Attributes*). The Origin entity includes the four asset origins under study in ORION, i.e. in-house development, outsourcing, open source, and COTS, whereas the Origin Suboptions details further aspects such as who the subcontractor is or if development is crowd-sourced. The Element Type and its sub-entities characterize the asset from a technical perspective, i.e. whether it is a software asset (source code), a system element (combination of hardware and software), an information element (document) etc. In total we have identified a set of 15 possible element types. Individual elements are then further detailed by Asset Attributes. We reuse these attributes from the ISO/IEC 25010 standard [26], covering 42 attributes, representing both functional aspects and qualities.

The Role Level relates with *Role Function* of a specific *Roletype* with a certain *Role Perspective*. This part of the ontology is used to describe the different people involved in the decision making process, e.g., requirements engineers, product managers, architects and senior developers. Combined with the possible perspectives, we hope that the ontology captures everyone involved in decisions regarding asset origins. Based on our initial studies of decision making in industry, we have identified 23 different role descriptions, and these were validated via the roles found in SWEBOK [27] and the BAPO model [28] (Business, Architecture, Process, and Organization).

The Decision Level describes whether a decision is strategic, tactical or operational, representing the scope and time frame of its effects. Furthermore, the Decision Level relates to of one

or more *Decision Criteria Types*: functionality, quality, time to market, financial and risk, and they are even more specified in the *Decision Criteria* entity. This entity represents the properties (e.g. effort, performance) that need to be evaluated to be used as important criteria in the decision process.

The final first-level entity in the ontology is the Method Family, consisting of different *Decision Methods*. We have identified four main families of decision support methods, for evaluating the criteria, partly based on work by Trendowicz and Jeferry [29]: 1) information from experts or past projects (data), 2) expert-based methods that rely on the expert judgments of one or more expert, 3) data-driven methods that depend on large amounts of data, and 4) hybrid methods that combine the first three methods. By reviewing literature, we have identified 55 different decision methods.

While our ontology shares many aspects with suggestions by other researchers, there are also differences. The main difference between our ontology and related works on ontologies and models for storing architectural knowledge and decisions is that we do not introduce explicit relationships between decisions from separate cases. Instead we aim at providing such cross-case comparisons in the knowledge repository of our decision support system by formulating queries to an underlying graph database, an approach further described in the following section.

V. Technical Considerations

A. Decision Support Systems in Software Architecture

Knowledge based decision support has been successfully exploited in areas as diagnostics support in the medical domain. It therefore seems worthwhile to investigate to what extent existing approaches from these domains are applicable also for the domain of software architecture decisions. However, contrarily to other domains such as medicine, experts in software development often provide better conclusions by themselves when compared to situations in which models or previous gained knowledge are used [30]. This comes from 1) a high amount of specific and tacit domain knowledge that is hard to capture in a model (e.g., that particular developers working in the project are likely to produce high quality code), and 2) the existing reasoning models do not accurately reflect the reality of software development due to the limited real-case studies they are based on. This contributes to a situation in which decision support in software development seldom capitalises on previous knowledge to derive new informed decisions.

The medicine field has a long tradition of using evidence-based and case-based reasoning to support the decision process and corroborate the conclusions. In particular, the use of decision support systems is based around a knowledge repository in order to reduce errors due to misinformed knowledge (e.g., bias, obsolete facts) and increase efficiency and quality of the decisions. The precondition is having a well-defined understanding of the attributes/characteristics/artefacts to record, as well as the processes to take decisions, which can be unambiguously digitalised [31].

Another domain which uses knowledge repositories is cognitive computing, which combines artificial intelligence with widely-available database sources to derive answers to specific questions [32]. In particular, the system relies on a knowledge corpus consisting of offline downloaded digital documents (e.g., Wikipedia, IMDB, dictionaries, textbooks, guidelines and manuals) and knowledge of previous cases to be able to provide the correct response to the provided clues.

Contrasting this, the domain of software architecture decisions is characterised by a much more limited set of cases contributing to the previous knowledge. Moreover, there is often no straightforward way to automatically gather detailed information from previous cases to build the repository, except explicitly requesting it to be manually entered. Some information could be automatically harvested from e.g., code repositories and bug reporting tools, but this is very limited when compared to, e.g., the medical domain, where much information is already available electronically for other purposes, e.g., in form of medical records.

B. Knowledge Storage, Retrieval, and Maintenance

The problem of storing and analysing big amounts of information is a research field per se [33]. Especially in the latest years, more and more software applications adapt their behaviour in order to fit a certain user profile, and most of the profiling mechanisms are based on knowledge collected and extracted through previous uses. The knowledge repository we envision for ORION shares some of the characteristics and issues faced in other data-intensive application domains, as discussed in the remainder of this section.

The exploitation of a knowledge repository can be enhanced by considering publicly available decision scenarios, since they help in building up the confidence about a certain choice (or the evidence, as we call it in ORION). On the one hand, companies could have all the interest in being part of such decision experience share, since others' experience could avoid erroneous choices. On the other hand, sharing decision problems opens up a lot of issues related to confidential information. Notably, a company would be reluctant in exposing weaknesses of current products, and in general to favour competitors to any extent. Therefore, for non disclosure reasons it could be necessary a two layer knowledge repository, one internal, possibly even with limited access to different roles in the company itself, and one external, in which publicly available cases are collected and can be used as evidence to support a certain decision. This is the case, for example, of airlines that own a private warehouse in which they store sensitive information about the company and passengers. In addition, airlines mine passengers' data and behaviour from publicly available information, such as airports statistics [34].

Another major issue to be taken into account when dealing with potentially big amounts of data is performances. In the case of ORION we envision the storage of disparate decision information that might be interconnected by means of multiple relationships, depending on each stakeholder's point of view, the application domain, the context, and so forth. In this

respect, trying to create a traditional relational database is not a practicable solution: in fact, the data schema would be based and optimised on a well-defined set of data queries, that would result in bad performances in all the other cases. More in general, querying a relational database would involve the join of large tables, which would turn out in long computations. Moreover, the underlying database schema would entail a well defined data entry procedure, which could result not suitable in different decision scenarios.

Therefore, we will opt for a graph database solution, in which access time is kept reasonably good on average (in most of the cases constant) [35]. Even more important, decision information can include arbitrary relationships (starting from the ontology defined by GRADE) that can be queried later on for building up the necessary evidence supporting a certain choice, which we consider as the distinguishing feature of the decision knowledge management offered by ORION. A potential drawback of this solution is that an appropriate data entry interface is needed in order to collect data from users and store them into the graph database accordingly. Such an interface has to be knowledgeable of what kind of relationships a stakeholder can state and update the repository correspondingly.

If documenting decisions is hampered by its time consuming consequences and low effectiveness, it is even less likely that stakeholders are willing to maintain decision information spontaneously [8]. In particular, ORION is interested in post-decision analysis, which might turn out as extremely relevant for future choices. In this respect, the knowledge repository shall provide adequate support for decision maintenance. An option could be to set "time-outs" for the stored knowledge, such that decision stakeholders are queried, after a certain period of time, about the outcome of a choice in the long run. In this case, a potential issue might be determining the appropriate life-span for an asset origin selection to have its measurable effects, since it greatly depends on the context in which the decision has been taken. Another possibility could be to exploit the interconnections existing in the knowledge repository itself, and let stakeholders declare sequences of interconnected decisions. In this way, when a new decision would be added to an existing sequence the system could automatically query the stakeholder about the reason, notably whether the subsequent decision was a consequence of the previous one(s) and in what sense (e.g., completion, refinement, fix, etc.) [8], [9].

VI. Conclusion

Decision knowledge collection and analysis is a research challenge: on the one hand, the amount of collected information should be enough to guarantee a certain level of confidence in future decisions as based of what is known about past cases; on the other hand, collecting, retrieving, and maintaining decision knowledge should be as transparent as possible to the user. In this paper we discuss the investigations done about providing a decision making process with an adequate knowledge repository support. In particular, in the ORION project we defined a decision process aiming at collecting the necessary evidence needed to justify a certain asset origin selection, among in-house, outsourcing, open-source, or COTS options. The management of decision information is done by means of a knowledge repository, which stores previous decision cases in terms of a well-defined ontology.

One of the peculiar characteristics of the solution we envision in this work is, indeed, the exploitation of a well-defined ontology that discloses the opportunity of drawing similarities with other decision cases, even pertaining to different system developments and in other applicative domains. Moreover, the exploitation of a graph database as technical implementation of the repository allows a more flexible storage and retrieval of decision knowledge.

Given the intrinsic characteristics of graph databases, the intended knowledge repository solution needs to be fed with decision cases in order to have a reliable validation of the proposal. In this respect, future work will be devoted to create the necessary user interfaces enabling data collection related to past decision cases. Moreover, it will be necessary to investigate deeper how to create a publicly available decision knowledge repository, in which relevant experience could be stored in a consistent and effective way.

Acknowledgment

This work is supported by a research grant for the ORION project (reference number 20140218) from The Knowledge Foundation in Sweden. We would also like to thank our colleagues in the ORION project for fruitful discussions that have helped improving the paper.

References

[1] P. Kruchten, P. Lago, and H. van Vliet, "Building up and reasoning about architectural knowledge," in *Proceedings of the Second International Conference on Quality of Software Architectures*, ser. QoSA'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 43–58.

[2] Z. Li, P. Liang, and P. Avgeriou, "Application of knowledge-based approaches in software architecture: A systematic mapping study," *Inf. Softw. Technol.*, vol. 55, no. 5, pp. 777–794, May 2013. [Online]. Available: http://dx.doi.org/10.1016/j.infsof.2012.11.005

[3] O. Zimmermann, L. Wegmann, H. Koziolek, and T. Goldschmidt, "Architectural decision guidance across projects - problem space modeling, decision backlog management and cloud computing knowledge," in *Software Architecture (WICSA), 2015 12th Working IEEE/IFIP Conference on*, May 2015, pp. 85–94.

[4] T. Vale, I. Crnkovic, E. S. de Almeida, S. Neto, Y. C. Cavalcanti, and S. R. de Lemos Meirad, "Twenty-eight years of component-based software engineering," *Journal of Systems and Software*, vol. 111, no. 1, pp. 128–148, January 2016.

[5] H. Breivold and M. Larsson, "Component-based and service-oriented software engineering: Key concepts and principles," in *33rd EUROMICRO Conference on Software Engineering and Advanced Applications, (SEAA)*, Aug 2007, pp. 13–20.

[6] "ORION: Decision-Support for Component-Based Software Engineering of Cyber-Physical Systems," 2016. [Online]. Available: http://www.orion-research.se

[7] A. Tang, P. Liang, and H. van Vliet, "Software architecture documentation: The road ahead," in *WICSA*, 2011, pp. 252–255.

[8] U. Zdun, R. Capilla, H. Tran, and O. Zimmermann, "Sustainable architectural design decisions," *IEEE Software*, vol. 30, no. 6, pp. 46–53, 2013.

[9] A. Tang, P. Avgeriou, A. Jansen, R. Capilla, and M. Ali Babar, "A comparative study of architecture knowledge management tools," *J. Syst. Softw.*, vol. 83, no. 3, pp. 352–370, Mar. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.jss.2009.08.032

[10] W. Ding, P. Liang, A. Tang, and H. Van Vliet, "Knowledge-based approaches in software documentation: A systematic literature review," *Inf. Softw. Technol.*, vol. 56, no. 6, pp. 545–567, Jun. 2014. [Online]. Available: http://dx.doi.org/10.1016/j.infsof.2014.01.008

[11] A. Jansen, A. Wall, and R. Weiss, "Techsure - a method for assessing technology sustainability in long lived software intensive systems." in *37th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 2011, pp. 426–434.

[12] M. Babar, T. Dingsøyr, P. Lago, and H. van der Vliet, *Software Architecture Knowledge Management: Theory and Practice*. Springer Berlin Heidelberg, 2009. [Online]. Available: https://books.google.se/books?id=Bj4wM_lRMx8C

[13] K. Schneider, *Experience and Knowledge Management in Software Engineering*. Springer Berlin Heidelberg, 2009. [Online]. Available: https://books.google.se/books?id=z_Gy6s63jE8C

[14] International Organization for Standardization, "ISO/IEC/IEEE 42010 Systems and Software Engineering - Architecture Description," ISO/IEC/IEEE, Tech. Rep., 2011.

[15] P. Kruchten, *Software Architecture Knowledge Management: Theory and Practice*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, ch. Documentation of Software Architecture from a Knowledge Management Perspective – Design Representation, pp. 39–57. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-02374-33

[16] A. Tang and H. Vliet, *Software Architecture Knowledge Management: Theory and Practice*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, ch. Software Architecture Design Reasoning, pp. 155–174. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-02374-39

[17] J. Tyree and A. Akerman, "Architecture decisions: demystifying architecture," *Software, IEEE*, vol. 22, no. 2, pp. 19–27, March 2005.

[18] R. Capilla, O. Zimmermann, U. Zdun, P. Avgeriou, and J. M. Küster, "An Enhanced Architectural Knowledge Metamodel Linking Architectural Design Decisions to other Artifacts in the Software Engineering Lifecycle BT - Software Architecture: 5th European Conference, ECSA 2011, Essen, Germany, September 13-16, 2011. Proceedings," I. Crnkovic, V. Gruhn, and M. Book, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 303–318. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-23798-033

[19] H. van Vliet, P. Avgeriou, R. C. de Boer, V. Clerc, R. Farenhorst, A. Jansen, and P. Lago, "The griffin project: lessons learned," in *Software Architecture Knowledge Management*. Springer, 2009, pp. 137–154.

[20] D. Falessi, G. Cantone, R. Kazman, and P. Kruchten, "Decision-making techniques for software architecture design: A comparative survey," *ACM Comput. Surv.*, vol. 43, no. 4, pp. 33:1–33:28, Oct. 2011. [Online]. Available: http://doi.acm.org/10.1145/1978802.1978812

[21] E. Kornyshova and R. Deneckère, *Conceptual Modeling – ER 2010: 29th International Conference on Conceptual Modeling, Vancouver, BC, Canada, November 1-4, 2010. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, ch. Decision-Making Ontology for Information System Engineering, pp. 104–117. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-16373-98

[22] E. Papatheocharous, K. Petersen, A. Cicchetti, S. Sentilles, S. M. A. Shah, and T. Gorschek, "Decision support for choosing architectural assets in the development of software-intensive systems: The GRADE taxonomy," in *1st International Workshop on Software Architecture Asset Decision-Making (SAADM)*, Springer, Ed., September 2015.

[23] T. Jewell, I. Anderson, A. Chandler, S. Farb, K. Parker, A. Riggio, and N. Robertson, "Electronic Resource Management - Report of the DLF ERM Initiative," Digital Library Federation, Washington, D.C., Tech. Rep., 2004.

[24] K. Petersen and C. Wohlin, "Context in Industrial Software Engineering Research," in *Proc. of the 3rd International Symposium on Empirical Software Engineering and Measurement*, 2009, pp. 401–404.

[25] M. Khurum, T. Gorschek, and M. Wilson, "The Software Value Map: An Exhaustive Collection of Value Aspects for the Development of Software Intensive Products," *Journal of Software: Evolution and Process*, vol. 25, no. 7, pp. 711–741, 2013.

[26] International Organization for Standardization, "ISO/IEC 25010 Software Engineering - Software Product Quality Requirements and Evolution SQuaRE Quality Model," ISO/IEC, Tech. Rep., 2008.

[27] P. Bourque and R. Fairley, "Guide to the Software Engineering Body of Knowledge, Version 3.0," IEEE Computer Society, Tech. Rep., 2014. [Online]. Available: www.swebok.org

[28] F. van der Linden, J. Bosch, E. Kamsties, K. Känsälä, and H. Obbink, "Software product family evaluation," in *Proc. of the 3rd Software Product Line Conference*, 2004, pp. 110–129.

[29] A. Trendowicz and R. Jeffery, *Software Project Effort Estimation - Foundations and Best Practice Guidelines for Success*. Springer, 2014.

[30] M. Jørgensen, "A review of studies on expert estimation of software development effort," *J. Syst. Softw.*, vol. 70, no. 1-2, pp. 37–60, Feb. 2004. [Online]. Available: http://dx.doi.org/10.1016/S0164-1212(02)00156-5

[31] A. Berlin, M. Sorani, and I. Sim, "A taxonomic description of computer-based clinical decision support systems," *Journal of biomedical informatics*, vol. 39, no. 6, pp. 656–667, 2006.

[32] R. High, "The era of cognitive systems: An inside look at ibm watson and how it works," *IBM Corporation, Redbooks*, 2012.

[33] "Challenges and Opportunities with Big Data. A community white paper developed by leading researchers across the United States," 2012. [Online]. Available: http://cra.org/ccc/wp-content/uploads/sites/2/2015/05/bigdatawhitepaper.pdf

[34] T. H. Davenport, "At the big data crossroads: turning towards a smarter travel experience," 2013. [Online]. Available: http://www.bigdata.amadeus.com/assets/pdf/Amadeus_Big_Data.pdf

[35] I. Robison, J. Webber, and E. Eifrem, *Graph Databases*. OReilly Media, 2015.