Analyzing Engineering Contributions using a Specialized Concept Map

Arnon Sturm[1,2], Daniel Gross[1], Jian Wang[1,3], Eric Yu[1]
University of Toronto[1], Ben-Gurion University of the Negev[2], Wuhan University[3]

sturm@bgu.ac.il, daniel.gross@utoronto.ca,
jianwang@whu.edu.cn, eric.yu@utoronto.ca

Abstract. Identifying open problems in an engineering domain is a first step towards making new contributions. To identify problems one often examines existing solutions to recognize opportunities for advances. As the knowledge in a domain grows and multiplies, it becomes increasingly difficult to keep track of advances made, especially in relation to evolving needs. Drawing from goal-oriented requirements engineering, we propose a specialized use of concept maps to map out contributions to problem-solving knowledge in an engineering domain. We illustrate the approach using the domain of Architecture Description Languages (ADLs) and discuss usefulness and usability of the specialized concept map.

Keywords: Knowledge Mapping, Requirement Engineering, Concept map

1 Introduction

The state of the art in engineering related fields is a fast moving target. With innovation occurring globally at fast pace, researchers and practitioners must expend significant efforts to keep up with the current state of the art, which is also a prerequisite to pushing the boundaries to better deal with new problems and needs.

To stay up-to-date and to make new contributions, researchers and practitioners must over time maintain an overview of a field, understand the problems addressed and solutions proposed, as well as identify the outstanding issues that should receive further attention. Given the fast pace of new developments, keeping such an overview is challenging. Researchers make use of a number of approaches to consolidate and better understand a research domain. They mainly use literature reviews, including systematic reviews [5], and tagging and classifications approaches[1] [6]. In addition, some research has been done to improve on aforementioned approaches such as to consolidate scholarly works using concept maps [8], cause maps [4], and claim-oriented argumentation [9]. As researchers are looking for innovation, they would like to have supporting tools that would help to cluster related topics, to explicate prob-

[1] There are also reference management systems like EndNote and Mendeley that support classification using folders and tagging.

lems and solutions, to represent and reason about differences between existing solutions, and to analyze how the knowledge in a domain evolves over time. Indeed, the aforementioned approaches offer different kinds of textual, conceptual and visual mapping over domains, however, analyzing their capabilities, it seems that they lack essential capabilities to evaluate or compare state-of-the art studies. Table 1 presents a (subjective) comparison of the approaches with respect to the needed capabilities.

Table 1. Comparing mapping techniques

Approach	**Clustering**	**Expressiveness**	**Reasoning**	**Dynamic Evolution**
Literature Survey	+	+	-	+
Classification	+++	-	-	+
Cause Map	++	++	++	+++
Concept Maps	-	+	-	+++
Argumentation	-	++	+++	+++

Based on our observation that contributions to the state of the art can often be characterized as *means* that come to address specific *ends* in some better way, and the lack of such a view in current approaches, we were motivated to seek an approach based on goal-oriented requirements engineering (GORE) approaches [11] in general and the i* (pronounced i-star) goal-oriented modeling framework approach in particular [11]. The i* approach has at its center the means-ends relationship, and the capability to differentiate alternate means towards some end by indicating their differing contributions towards desired quality objectives. Following that approach and based on our previous work [3], we propose a knowledge mapping technique to represent and map out problems and solutions in engineering domains. We envision that using such a technique would better support researchers in representing and reasoning about research advancements in engineering domains. In this paper, we describe the technique, illustrate its use for the domain of software architecture description languages (ADLs), present initial evaluations and further discuss our vision in developing the technique.

In Section 2 we briefly review the ADL landscape and point to some of the issues we observed. In Section 3 we define the mapping technique and demonstrate it using examples from the ADL domain. In Section 4 we further discuss the proposed technique and reflect on its use. In Section 5 we conclude and elaborate on our vision regarding the usage of such knowledge maps.

2 The Architecture Description Language Domain

The motivation for our research is the observation that much of the knowledge underpinning engineering domains can essentially be characterized using means-ends relationships and qualifying properties. Such a characterization supports systematically representing needs and problems and their linkages to proposed solutions that come to

address those problems in some better way. Capturing problems, solutions and such linkages between them helps in systematically identifying what the problems in a domain are, which of those have been addressed to date and how, as well as what the outstanding issues are that could benefit from future exploration.

As an example we looked at the domain of architectural design as one of the important areas in IS design. Specialized languages for supporting architectural design has been an active research area for some time, yet their industry adoption has been limited. In a recent survey that aimed to identify what architects need from architectural description languages (ADLs) [7], the authors identified 150+ ADLs that were proposed over the last decade and half and asked practitioners, amongst others, which ADLs they used, what ADL features they found they need, and more generally what their ADL needs are during architectural development.

One surprising outcome of the survey was the limited adoption of the proposed ADLs in industry. Only a hand-full of ADLs, and mainly those that originated from industry, were in active use, with UML used as an ADL by 86% of the respondents. Furthermore, 86% of respondents indicated that ADLs needed to be extended to meet project-specific needs. Yet, only about 25% of UML users actually extended UML, such as by use of profiles, to meet project-specific needs, with about 73% of respondents using UML as-is, despite its lack of architectural description features.

To better understand the reason of UML adoption, and in particular why it was mostly adopted in organizations as-is without extensions, we mapped out portions of the ADL domain to capture how ADL authors perceived problems and solutions in that domain. Our aim was to identify how a knowledge structure based on means-ends relationships could more systematically tie ADL research to practitioners' needs. While addressing architects needs is clearly an important objective for ADL designers, creators of ADLs nevertheless typically focused on specific technical features they perceived architects would need and offered interesting representational or analytical features, which in the end were however not adopted by practicing architects in industry.

To further examine knowledge structures in the ADL domain we also turned to ADL literature reviews, which consolidate, compare and contrast ADLs. Such literature surveys offer useful perspectives on the knowledge structure of the ADL domain. However, such reviews mainly focused on comparing the feature sets of ADLs using predefined features or feature categories perceived to be of value to the survey authors [2].

We thus aim to complement such textual survey approaches with a conceptual knowledge map that helps characterize and clarify the problems and needs, and different solution approaches in engineering domains, and helps link high level needs and problems to the solution approaches.

3 Specializing Concept Maps for Engineering Domains

During our research we applied the proposed technique to map out portions of a variety of engineering domains including agent-oriented software engineering, geo-

engineering, web mining, and documenting software architectures. Distilling our modeling experience we adopted a minimal set of modeling constructs including two main types of nodes and several types of links. By convention, the map is laid out with problems or objectives at the top and solutions at the bottom.

- The **task** is the main element in the means-ends hierarchy. A task can be interpreted either as a problem (in relation to lower elements) or a solution (in relation to higher elements). It is named with a concise verb phrase typically, and graphically depicted as a rectangular shape with rounded corners. For example, in Figure 1, which illustrates part of the ADL knowledge map, the task "Define Architecture" is a problem to be addressed. It can be achieved by the tasks of "Define New Architecture" or "Utilize Existing Knowledge". Both solutions can in turn can also be viewed as sub-problems that need further addressing. A task can have associated contexts and a set of references (i.e., the knowledge sources) justifying the existence of such a task within the map. These are not shown in Figure 1 to avoid clutter, but are supported by the tool.
- A **quality** element is used to express quality attributes that are desired for associated tasks. Examples in Figure 1 include "Scalable", "Traceable", and "Architecture Quality". A quality is depicted as an ellipse, and is typically named with adverbial or adjectival phrases or quality nouns (e.g., "-ilities").
- Links connect tasks and qualities. In the following we elaborate on the link types.
  - The **achieved by** link represents a means-end relationship. The arrow points from the "end" to the "means". Figure 1 indicates that "Use ADL" is one way of achieving "Design New Architecture". "Use WRIGHT" and "Use UML" are alternative ways of achieving "Use ADL".
  - The **consists of** link indicates that a task has several sub-parts, all of which should be accomplished for the parent task to be accomplished. In Figure 1, "Devise Architecture" consists of "Define Architecture", "Select Technology", and "Communicate the Architecture", among other problems that need to be addressed.
  - The **association** link (an unlabeled and non-directional link) indicates the desirable qualities for a given task. These qualities are to be taken into account when evaluating alternative ways for accomplishing the task. For example, "Adoptable", "Extensible", are qualities that could differentiate among different ways of "Use(-ing) ADL".
  - The **extended by** link indicates that the target task is an extension of the source task. For example, "Create UML-Profile" is an extension of "Use UML". All qualities that hold for the parent task also hold for its extensions.
  - The **contribution** link (a curved arrow) indicates a contribution towards a quality, from a task or another quality. The contribution can be from strong negative to strong positive contribution, which are determined subjectively by the map creator based on the available resources. For example, "Use UML" is well contributed ("++") to "Weavable into SDLC", and also contribute ("+") to "Analyzable". The alternative "Use WRIGHT" is well contributed ("++") to "Formal" and contribute ("+") to "Weavable into SDLC", and "Usable".

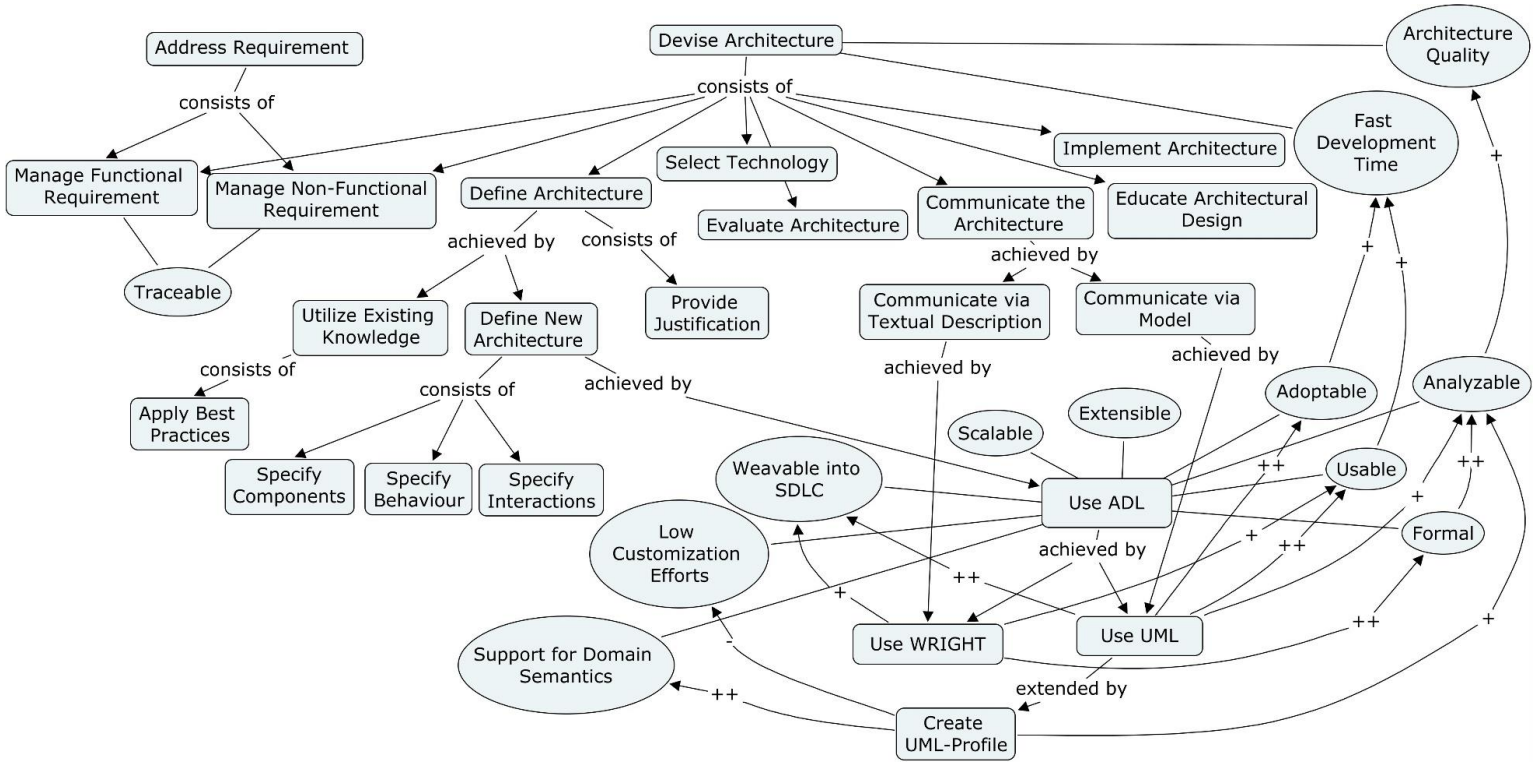

Figure 1. A partial knowledge map of the ADL domain

We used the cmap tool[2] to draw the knowledge map in Figure 1. Using the cmap tool allows us to benefit from all implemented features of the platform, including collaborative modeling and sharing of concept maps.

It should be noted that Figure 1 is a map aims to serve as an index to the actual knowledge. The purpose of the map is not to create new knowledge rather to organize the knowledge in a way that would increase its accessibility.

To construct the map in Figure 1, we analyzed an informal survey of architects duties and skills offered on the CMU SEI website [9] and the aforementioned ADL architects needs survey paper [7]. We further looked at comparison of various ADLs reported in the literature [2]. Following these resources, we were able to construct the map while having supporting evidence for the claims implied by nodes and relationships included in map.

4 Discussion

Using the knowledge map in Figure 1 we can point to the main justifications for choosing one ADL over another by following their level of contribution to the qualities associated with the use of an ADL. Which ADLs were chosen in practice, and why? According to the Malavolta survey [7], most architects adopting UML as an ADL decided to live with the limitations of UML rather than choosing an ADL that fits with their specific software architectural analysis needs, because the generic version of UML was a notation already known in the organization, and hence required minimal learning effort during adoption. This explanation is indicated in Figure 1 by the contribution link from "Use UML" to "Analyzable": using UML has limited contribution to having an analyzable architectural description; a "++" link from "Use UML" to Adoptability: using UML has a strong contribution to being adoptable in the organization; a "++" link from "Use UML" to "Weaveable into SDLC": indicating another advantages over WRIGHT; and the "-" contribution link from "Create UML Profile" to "Low Customization Efforts", indicating that creating a UML Profile contributes negatively to reducing customization efforts, and thus rarely used.

Hence, the concept map helps capture and visualize, using selected concepts and relationships, the arguments that drove many architect's adoption decision-making. More specifically, adoptability in the organization is an overriding concern, which needs to be addressed well before other useful features, are introduced in an ADL.

Another insight for future research that may be derived from the knowledge map is that ADLs only cover a small part of the overall responsibilities of software architects. According to Figure 1 decisions about ADL use mainly support the definition of new architectures, which is, however, only one task of a variety of others that architects are engaged in during architectural development, some of which might be more important to the organization than more formal descriptions of architectures. The knowledge map should thus help researchers and practitioners in seeing the broader picture of needs into which more particular research directions are positioned.

[2] http://cmap.ihmc.us/

While the proposed technique facilitates representing problems and solutions in existing state-of-the-art, we encountered a number of challenges while self-examining it in the various domains aforementioned and with various viewpoints (e.g., mapping a domain, mapping a specific research, adopting a top-down approach, and adopting a middle-out approach):

Conceptual Mismatch: Identifying problems, solutions, qualities, and the relationships among them is often non-trivial. Researchers and stakeholders often present needs and benefits in solution-oriented terminology and languages and neglect the connection with the problem-oriented aspect.

Naming Decompositions: During the construction of a knowledge map elements are decomposed into lower level elements. Decomposition is the main mechanism to unearth variation and differences in approach details (solution features) that matter with respect to qualities. However, in some domains it appears difficult to identify and name those solution feature "components" that differentiate among alternative approaches. This suggests that more holistic representations of solution approaches, or, that more finer-grained concept map based analysis guidelines to help make explicit in what way proposed solutions differ in their details, may be needed.

Multiple vantage points and terminology use: Because of different viewpoints map creators might take, they may develop maps differently, both in terminology and in the abstraction level. Furthermore, it is in the purview of the map creator to decide which level of abstraction is the most fitting to express problems and solution approaches. When constructing larger maps out of contributions from different map authors, aligning the levels of abstraction is thus non-trivial.

Scalable tool support: Having good tool support is often a key weakness in proposed approaches. Using concept maps we can take advantage of existing tools, and use "scalability" features such as: element expanding/collapsing and map referencing.

Domain knowledge extraction: Currently, knowledge extraction and its mapping are done manually and obviously, subjectively, as implied before. This introduce a burden on adopting the approach. Nevertheless, we envision crowd-mapping as an approach that distributes the burden across interested many participants, who benefit from mutual contributions, and approaches to automated concept extraction from bodies of engineering text guided by the proposed concepts that link needs with solutions.

5 Conclusion and Future work

With the fast moving technological/engineering innovations landscape, new approach proposals that address novel challenges are continuously devised. In this paper we propose a technique to map out research fields using a light-weight modeling technique, based on a well-known concept mapping approach and argue for its benefits, such as the ability to represent and facilitate the analysis for novel proposals, gaps of un-addressed problems, as well as, other kinds of analysis such as tracing of possible reasons for adoption or non-adoption of proposed approaches. We believe that the approach is applicable to any engineering domains that fit into the problem-solution means-end approach. Yet, its benefits depend on the domain maturity.

From preliminary user studies [1], we found that the proposed technique supports examining a domain of research not only from the solution point of view, but also allows for emphasizing problems addressed, and in particular the properties in the problem space that offer significant advantages over prior art. Having both problems and solutions captured in one place helps better reviewing and understanding a domain. We were also able to identify gaps in which further research could be performed.

To further explore and facilitate the use of knowledge mapping, in the future we plan to expand knowledge map capabilities in a number of directions: further develop guidelines for map creators to support extracting knowledge from research domains and including them in knowledge maps; support a crowd-mapping approach whereby different stakeholders could contribute to creating, arguing about and improving on a collaborative knowledge map; support for trust mechanisms, as well as, empirical evidence based additions to knowledge maps that offer additional insights; develop semi-automated reasoning support to identify gaps or even possible solution approaches to identified gaps, across different knowledge maps; and develop automated extraction of knowledge mappings from bodies of engineering texts, guided by core concepts proposed in this paper. Additional evaluations for testing the benefits of the proposed technique are also required.

References

1. Abrishamkar, S.: Goal-Oriented Know-How Mapping, Modelling process documentation, a prototype, and empirical studies, M.Sc. Thesis, University of Toronto (2013)
2. Clements, P.C.: A survey of architecture description languages. Proceedings of the 8th International Workshop on Software Specification and Design, pp.16-25 (1996)
3. Gross, D., Sturm, A., Yu, E.: Towards Know-how Mapping Using Goal Modeling. iStar, 115-120 (2013)
4. Eden, C., Ackermann, F., Cropper, S.: The analysis of cause maps. Journal of Management Studies, 29(3), 309–324, (1992)
5. Kitchenham, B., Brereton, P., Budgen, D., Turner,M., Bailey,J., Linkman, S.: Systematic literature reviews in software engineering - A systematic literature review. Inf. Softw. Technol. 51 (1), 7-15 (2009)
6. Kwasnik, B.: The role of classification in knowledge representation and discovery. Library Trends, 48, 22-47 (1999)
7. Malavolta, I.,Lago, P., Muccini, H.,Pelliccione, P., Tang, A.: What Industry Needs from Architectural Languages: A Survey, IEEE Transactions on Software Engineering, 39 (6), 869-891 (2013)
8. Novak, J. D., Cañas, Al. J.: The Theory Underlying Concept Maps and How To Construct and Use Them, Institute for Human and Machine Cognition (2006)
9. SEI, http://www.sei.cmu.edu/architecture/research/previousresearch/duties.cfm, Shum, S.B., Motta, E., Domingue, J.: ScholOnto: An ontology-based digital library server for research documents and discourse. International Journal of Digital Libraries, 3(3): 237-248 (2000)
10. Yu, E., Giorgini, P., Maiden, N., Mylopoulos, J.: Social Modeling for Requirements Engineering, MIT Press (2011)