Continuous Architectural Knowledge Integration: Making Heterogeneous Architectural Knowledge Available in Large-Scale Organizations

Juergen Musil*, Fajar J. Ekaputra*, Marta Sabou*, Tudor Ionescu†, Daniel Schall†, Angelika Musil* and Stefan Biffl*
*Institute of Software Technology and Interactive Systems
Vienna University of Technology, Vienna, Austria
Email: {firstname.lastname}@tuwien.ac.at
†Scalable and Resilient Architectures
Siemens AG, Vienna, Austria
Email: {firstname.lastname}@siemens.com

***Abstract*—The timely discovery, sharing and integration of architectural knowledge (AK) have become critical aspects in enabling the software architects to make meaningful conceptual and technical design decisions and trade-offs. In large-scale organizations particular obstacles in making AK available to architects are a heterogeneous pool of internal and external knowledge sources, poor interoperability between AK management tools and limited support of computational AK reasoning. Therefore we introduce the Continuous Architectural Knowledge Integration (CAKI) approach that combines the continuous integration of internal and external AK sources together with enhanced semantic reasoning and personalization capabilities dedicated to large organizations. Preliminary evaluation results show that CAKI potentially reduces AK search effort by concurrently yielding more diverse and relevant results.**

***Index Terms*—Architectural knowledge management, continuous software architecture, semantic integration.**

I. Introduction

The organization and management of *architectural knowledge* (AK) have become an important aspect of software architecture at management and technical levels alike [1]. In particular, the timely discovery, sharing and integration of AK [2] is critical for enabling software architects and developers to make conceptual and technical design decisions and trade-offs [3]. Capilla et al. [4] identified internal and external factors that influence the use of AK amongst practitioners. Beyond the challenges of creation, maintenance and traceability of AK [4], there are also integration issues. Specifically, the integration of architectural knowledge management (AKM) systems along with existing general-purpose knowledge management systems (such as enterprise-grade wikis) and their underlying models represent further obstacles that hinder the introduction and adoption of AK-specific tools and knowledge repositories in organizations. This results in cumbersome workflows and information redundancy [4].

In particular the challenges are twofold. Firstly, architects and developers do not always document and share AK in organization-internal systems, but for the sake of convenience just refer to a diverse collection of organization-external information sources (e.g., blog posts, Q&A portals, GitHub repositories). This circumstance undercuts the adoption of comprehensive AKM approaches which are often perceived by practitioners as overly prescriptive in the modes and means of how knowledge ought to be organized. Secondly, existing AKM tools and approaches often tend to assume a greenfield scenario when it comes to knowledge management. As a consequence, existing AKM tools do rarely consider how they might collectively work together with other third party AKM tools, since they are tentatively conceptualized as "silos" with limited openness and interoperability. In combination, the aforementioned limitations hinder the effective utilization of AK amongst practitioners as well as the creation of support for sustainable AKM processes.

In order to address these challenges we propose the *Continuous Architectural Knowledge Integration (CAKI)* approach, which enables the integration and interoperability of internal and external knowledge sources and AKM tools in large-scale organizations. Our work has two main contributions: (1) a pipeline for the continuous AK acquisition and integration based on tailored, and granular semantic AK models, and (2) improved AK discoverability through enhanced computational reasoning and personalization capabilities. Furthermore, CAKI facilitates software architects the efficient discovery and gradual exploration of relevant architectural knowledge from various knowledge sources through a unified point of access inspired by the simplicity of Google's minimalistic search interface.

In a first run, we evaluated the utility and effectiveness of the approach through a pilot implementation at Siemens, a company which conducts large-scale software development projects. Our preliminary results indicate that the CAKI approach produces more relevant results for architects than traditional, generic approaches like enterprise search, and improved discoverability of content in AK repositories. In particular, the capability of faceted exploratory search enabled by CAKI

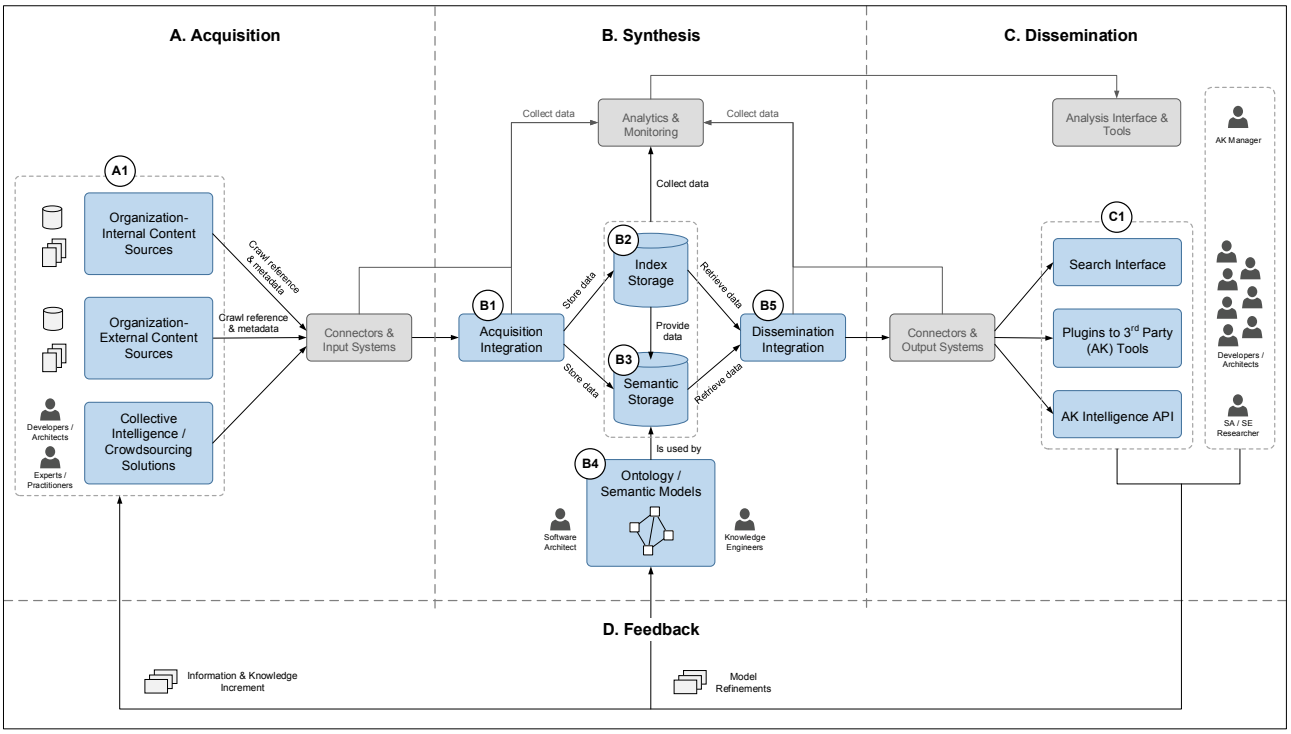

Fig. 1. Continuous Architecture Knowledge Integration (CAKI): Multi-sided AK integration and discovery consisting of Acquisition, Synthesis, Dissemination and Feedback.

allowed architects a fine-grained, incremental exploration of AK content.

The remainder of this work is structured as follows. Section II presents an overview of the CAKI approach comprising four processing stages. Preliminary results in form of a pilot study are reported in section III. Section IV discusses related work and section V concludes and outlines future work.

II. Continuous Architectural Knowledge Integration Approach

In this section we present the Continuous Architectural Knowledge Integration (CAKI) approach[1], which enables (1) the sustained integration of heterogeneous, organization-internal and external knowledge sources, and (2) more refined reasoning capabilities. CAKI realizes a knowledge integration pipeline, that consists of four stages (see Fig. 1): The *Acquisition* stage actively acquires information from content sources, then *Synthesis* stage integrates the data using semantic technologies and organization-specific, semantic AK models. *Dissemination* stage supplies an AK search interface, as well as AKM-specific and general-purpose KM tools with filtered and personalized AK content. Finally, the *Feedback* stage uses meta data from the integrated content and user activity data to continuously adapt the semantic AK models and acquisition strategies. In the following paragraphs we describe the individual stages in more detail.

[1] *http://qse.ifs.tuwien.ac.at/proj/caki/* (last visited 2017-02-28)

A. Acquisition Stage

The Acquisition stage provides a means for acquiring information from different Content and Information Sources (A1). These sources can be either organization-internal (e.g., enterprise wikis, AK repositories, AKM tools) or external (e.g., websites, blogs, archival publications) and are automatically indexed by the system. Information acquisition includes explicit and tacit knowledge, whereby tacit knowledge might be manually codified by architects, developers and domain experts using crowdsourcing and collective intelligence systems, as suggested in [5]. External sources are crawled using Google Custom Search bindings.

B. Synthesis Stage

The Synthesis stage consolidates the acquired AK by (1) extracting content indexes, which are used for full-text AK content search, and (2) adding semantic information to allow intelligent content reasoning, personalization and fast exploratory search.

Acquisition Integration (B1) filters AK content, creates indexes, and assigns semantic information according to the predefined semantic AK models. The storage system consists of two components: First, an Index Storage (B2) stores content indexes. Second, a Semantic Storage (B2) retains semantic information about the data, in particular weighted relations between AK content and semantic models.

The Semantic AK Models (B4) are relevant for concept integration and contain AK concepts, relations, and constraints. The models provide the basis for automated inference of

additional AK relations (e.g., to derive relations between two AK concepts based on their properties similarity). Furthermore the models represent the organization's AKM "working model" and thus are designed to integrate AK across organizational boundaries (business units, research groups) and are key enablers of dissemination and reuse. In order to satisfy concerns with software architecture and knowledge management, models are expected to be a joint work product of software architects and knowledge engineers. Model tailoring can be done in three granularity levels: generic, organization-specific, and domain-specific. Generic level includes models comparable to ISO/IEC/IEEE 42010 meta models [6]. At the organization-specific level, models are adapted to concepts and standards internal to the organization. Finally, domain-specific granularity level refines organization-level models with respect to a particular application domain. Each granularity level increases the degree of model detail, and subsequently improves reasoning precision of the CAKI approach, but also increases upfront modelling and model maintenance effort. Dissemination Integration (B5) intelligently combines the query results from Index Storage (B2) and Semantic Storage (B3) and personalizes the result sets with respect to individual user and attached client type.

C. Dissemination Stage

The Dissemination stage provides software architects and developers with access to AK via different sophisticated interfaces and connected tools (C1). Below we describe representative examples of such CAKI-enabled interfaces and tools:

1. Exploratory-Faceted AK Search: Users can iteratively explore and receive personalized AK results that are relevant to them.

2. Plugins to 3rd party (A)KM tools: Filtered query results can be integrated and accessed within other KM tools and thus overcomes limited openness and interoperability between tools.

3. AK Intelligence API: This API provides (A)KM tools an interface to the reasoning, personalization and context inference services of a CAKI system. This enables (A)KM tools to develop novel capabilities by relying on organization-wide consolidated architectural intelligence. The interface also provides access to data analytics capabilities that allow AK managers to assess the needs for future CAKI improvements and other knowledge management activities.

D. Feedback Stage

The Feedback stage feeds quantitative and qualitative AK usage information from the Dissemination stage back to the Acquisition and Synthesis stages for self-adaptation. In the Synthesis stage, usage data is gathered for steering evolution and the continuous self-adaptation of semantic AK models. It thus reduces the effort to keep models up to date. Subsequently this mechanism would enable the realization of the concept of *Liquid Models* [7] in the context of architectural knowledge. In the Acquisition stage, usage data is used to adapt acquisition strategies (e.g., less popular sources are less frequently updated).

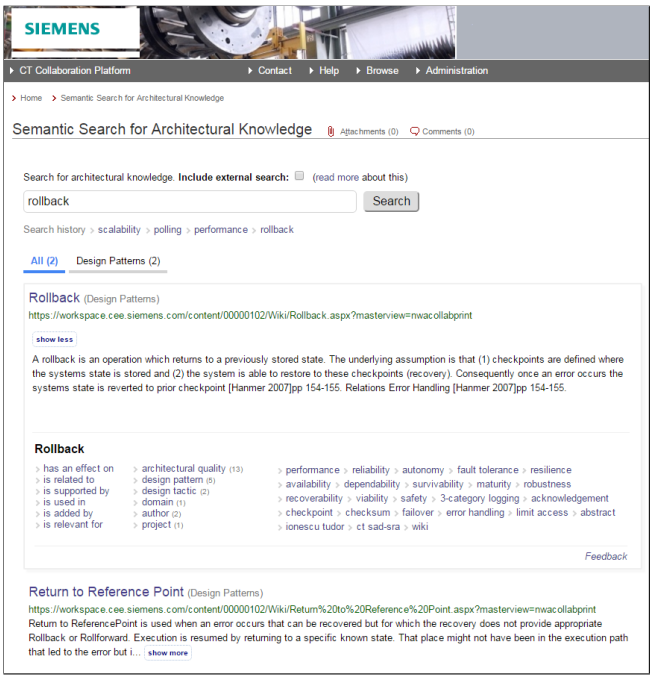

Fig. 2. CAKI-generated results within an exploratory search interface closely integrated into the existing knowledge management platform.

III. Preliminary Results

We performed a pilot evaluation of the CAKI approach in the context of a real-world scenario within Siemens AG, a large-scale organization. Since our approach assumes a brownfield starting scenario, the major selection criteria for a suitable candidate organization were that it has a high baseline of AKM practice, methods and routinely uses a diverse set of (A)KM tools across the company.

In the Siemens scenario an index storage (B2 in Fig. 1) was already in place to support software architectural knowledge and expert search. However, the AK search process was still challenging for users due to scattered knowledge sources and limited semantic annotation attached to the knowledge of this scattered data. These challenges lead to limited search results and unclear relations between each search result.

For the pilot, we created generic and organization-specific semantic models (B4 in Fig. 1). Based on these models, semantic AK information (e.g., related concepts and types of relation between concepts) and its context (e.g., knowledge authors, relevant projects and user roles) were extracted and integrated within the semantic storage (B1, B3 in Fig. 1). For the acquisition stage we used external web sources connected through Google Custom Search and a small set of internal AK repositories and wikis. For the dissemination stage we focused on exploratory-faceted search integrated into the Siemens AK Search results, as depicted in Fig. 2. In this scenario, users can search for software architecture knowledge (e.g., specific tools, methods, design patterns and tactics) as well as experts. For

each result, the system provides users with relevant related concepts and their relation to the selected query element. Users can also conduct faceted filtering by selecting one or more relations and/or types of related concepts in the result box. Additionally, the search history is persisted and displayed beneath the search-box. This facilitates fast repeated searches and provides users with breadcrumb path back to their initial search interests, while also enabling knowledge engineers to better understand search patterns through data analytics.

One of the main goals of the pilot was to guide users during their search process toward suitable software design methods related to different architectural concepts and design patterns. Pilot trials with architects and developers showed that the CAKI approach yields richer and more relevant results compared to search capabilities of existing wikis and other search systems at enterprise level. Further results also indicate a more efficient search process and improved discoverability of AK in repositories that are only poorly integrated into the current corporate knowledge management landscape.

IV. Related Work

Capilla et al. [4] provide an overview of the evolution of software architecture knowledge management tools and identify a trend towards tools that provide more sophisticated reasoning capabilities. Their overview shows, indeed, that reasoning is only sparsely used currently and primarily serves the task of inconsistency detection. Similarly, personalization capabilities in existing tools are limited. In our work, we rely on ontology-based models to allow such sophisticated reasoning as a basis for personalization mechanisms and exploratory search. The ability to perform reasoning is a characteristic of knowledge-based approaches, which are widely used in software documentation in general and the management of software architecture documents in particular [8]. Ontology-based technologies were used by leveraging on recent developments in the Semantic Web research area, including web-based knowledge representation languages such as RDF(S) and OWL, with strong support for reasoning. For example, De Graaf et al. [9] present an ontology-based approach to search for AK and an experiment to compare it with the traditional, file-based search. They propose a wiki-based system where the underlying ontology provides a backbone for organizing information in a more structured way. The ontology enables more structured navigation for finding information, faceted search based on concept properties, and the execution of pre-defined SPARQL queries. Ontologies and semantic-wikis are the basis of the Ontobrowse approach proposed by [10] for managing the documentation of Service-Oriented Architectures. The underlying ontology model allows better browsing, searching, and querying documents. Similarly, Tang et al. [11] present an approach to search for AK that employs a combination of a light-weight ontology and a wiki-based system. Finally, ontologies are used as a basis for intuitive, graphical visualizations of AK in [12]. To conclude, several studies have investigated the use of ontologies to support the retrieval of AK documentation. Primarily light-weight ontologies were used which were often combined with wiki-based systems offering support for browsing, faceted-search and querying. Yet, these systems still fall short of exploratory search and personalization mechanisms that rely on sophisticated reasoning algorithms.

V. Conclusion and Future Work

In this paper we introduced with CAKI a novel AKM approach that combines the continuous integration of organization-internal and external AK sources together with enhanced semantic reasoning and personalization capabilities. The CAKI approach demonstrated its potential in an initial evaluation by showing a reduced AK search effort with concurrently yielding more diverse and relevant results. As future work we plan to (1) extend the information sources/targets in the acquisition and dissemination stages, (2) improve AK model adaptation capabilities and (3) conduct a broader evaluation with business units from different application domains. We think that CAKI introduces a new perspective into AKM approaches when it comes to knowledge integration, which might also inspire other approaches in this direction in the near future.

References

[1] P. Kruchten, P. Lago, H. Van Vliet, and T. Wolf, "Building up and Exploiting Architectural Knowledge," in *5th Working Conf. on Software Architecture (WICSA '05)*. IEEE Computer Society, 2005, pp. 291–292.

[2] R. C. De Boer, R. Farenhorst, P. Lago, H. Van Vliet, V. Clerc, and A. Jansen, "Architectural Knowledge: Getting to the Core," in *3rd Int'l Conf. on Quality of Software Architectures (QoSA '07)*. Springer Berlin Heidelberg, 2007, pp. 197–214.

[3] S. Stevanetic, K. Plakidas, T. B. Ionescu, F. Li, D. Schall, and U. Zdun, "Tool Support for the Architectural Design Decisions in Software Ecosystems," in *Europ. Conf. on Software Architecture Workshops (ECSAW '15)*. ACM, 2015, pp. 45:1–45:6.

[4] R. Capilla, A. Jansen, A. Tang, P. Avgeriou, and M. A. Babar, "10 years of software architecture knowledge management: Practice and future," *Journal of Systems and Software*, vol. 116, pp. 191–205, 2016.

[5] J. Musil, A. Musil, and S. Biffl, "Introduction and Challenges of Environment Architectures for Collective Intelligence Systems," in *Agent Environments for Multi-Agent Systems IV*, ser. LNCS. Springer International Publishing, 2015, vol. 9068, pp. 76–94.

[6] ISO/IEC/IEEE 42010, *Systems and Software Engineering - Architecture Description*, 2011.

[7] A. Mazak and M. Wimmer, "Towards Liquid Models: An Evolutionary Modeling Approach," in *18th IEEE Conf. on Business Informatics (CBI '16)*, 2016, pp. 104–112.

[8] W. Ding, P. Liang, A. Tang, and H. Van Vliet, "Knowledge-based approaches in software documentation: A systematic literature review," *Information and Software Technology*, vol. 56, no. 6, pp. 545–567, 2014.

[9] K. A. De Graaf, A. Tang, P. Liang, and H. Van Vliet, "Ontology-based software architecture documentation," *Joint 10th Working Conf. on Software Architecture and 6th Europ. Conf. on Software Architecture, (WICSA/ECSA '12)*, pp. 121–130, 2012.

[10] H.-J. Happel, S. Seedorf, and M. Schader, "Ontology-enabled Documentation of Service-Oriented Architectures with Ontobrowse Semantic Wiki," in *PRIMIUM - process innovation for enterprise software*, ser. GI-Edition, vol. 151. Ges. für Informatik, 2009, pp. 61–80.

[11] A. Tang, P. Liang, and H. Van Vliet, "Software Architecture Documentation: The Road Ahead," in *9th Working Conf. on Software Architecture (WICSA '11)*. IEEE, 2011, pp. 252–255.

[12] P. Kruchten, P. Lago, and H. Van Vliet, "Building Up and Reasoning About Architectural Knowledge," in *2nd Int'l Conf. on Quality of Software Architectures (QoSA '06)*. Springer Berlin Heidelberg, 2006, pp. 43–58.