

Co-evolving Digital Architecture Twins

Sven Jordan¹

¹Group IT Solution & Enterprise Architecture, Volkswagen AG, 38440, Germany

Abstract

Software development in industry is getting increasingly complex as systems are getting more sophisticated and are often interconnected constituting the system landscape. Architecture description is therefore getting increasingly important. The necessary maintenance of description is often neglected because of different priorities due to time and budget constraints. This leads, among other things, to outdated architecture description. For a more efficient planning of the architectural landscape and prevention of redundancy, it is vital that architects and other stakeholders have the most current information about the systems. This paper presents doctoral research in its early stages concerned with the issue of continuous architecture recovery allowing to reflect the current architecture and evolution of the system as a digital architecture twin. The proposed approach aims to automatically extract architecture information of complex systems by recovering it from heterogeneous architectural data sources. The idea is the consolidation and integration of this recovered architecture information at different points in time to enable the representation of the system and its evolution. This permits the use of an architecture information query language facilitating different use cases (e.g., support of architectural design decisions or tailored architecture description). Planned contributions are the assessment and consolidation of heterogeneous information sources and the application of architecture recovery methods with the noteworthy addition of versions over time of those information sources and the creation of a co-evolving digital twin.

Keywords

Architecture recovery, Digital twin, Architectural design, System landscape recovery

1. Introduction and problem statement


One of the main problems in software architecture evolution and maintenance is the low quality and even non-existence of architecture description (e.g., architecture models) as systems evolve and increase in complexity, and are adapted to environments, technology or customer requirements. Evolution of a system should entail evolution and maintenance of its description, as otherwise the description does not reflect the actual system anymore, resulting in a decrease of quality and usefulness of the architecture description. Yet, the creation and maintenance of architecture description is linked with high effort (time and costs) as it is a primarily manual task. However, an updated description is a key driver for an architect to understand a system, comprehend dependencies and decide on future enhancements. Furthermore, stakeholders require different views [1] on a system at different levels of abstraction or granularity. It adds to the effort to keep the quality of and the description itself consistent considering the different views and abstraction levels resulting in even higher costs. To counter the problem of

ECSA'21: 15th European Conference on Software Architecture, September 13–17, 2021, virtual

✉ sven.jordan@volkswagen.de (S. Jordan)



© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

orphaned documentation and decreasing description quality, software architecture recovery [2] methods are used. Architecture recovery is referred to as methods and processes for retrieving architecture information from an implemented system and associated data sources. This recovered architecture information reflects the current state of a system. However, as the system evolves, so does the architecture (information). This leads to a need for a continuous process of architecture recovery, which considers heterogeneous data sources and versions over time to reflect the system as accurately as possible. We intend to automatically recover, consolidate and integrate architecture information from heterogeneous data sources. This architecture information will be mapped into a unified architecture information model, which we consider a digital architecture twin representing a system. As the system evolves, this digital twin needs to continuously co-evolve with the system. We further intend to develop an architecture query language able to retrieve architecture information from the digital architecture twin to support architecture design decision making, the identification of prevailing architectural patterns or the creation of tailored architecture description.

2. State of the art and open challenges

Automated architecture recovery approaches range from static to dynamic methods using different techniques like structural clustering, concern-based clustering, or interactive exploration to extract and recover different layered architecture information and employing input parameters like the implemented system (e.g., as source code). Approaches like ACDC [3], WCA [4] or LIMBO [5] belong to the clustering methods retrieving clusters representing subsystems based on structural information. Concern-based approaches like ARC [6] or RELAX [7] add concerns to the clustering approach yielding precise and comprehensible clusters with context. Evolution-based approaches consider the evolution of a software system (e.g., using source code or issue management tools) taking a system's legacy into account [8] to recover architecture design decisions. Interactive methods like the Grounded Theory approach [9] focus on a more general approach to recover architecture information. It is described as a human intensive and relatively costly process, which has the benefit of being as general as possible, therefore applicable to almost every system. Proposed workbench approaches are Rigi [10] or ARCADE [11]. These approaches enable interactive exploration as they extract data and reconstruct architecture information and architecture views of a system. Even though these methods produce valuable architecture information, they tend to be laborious, require manual effort, or operate on single viewpoints of a system. These are open challenges for software architecture recovery: (1) the identification of possible data sources in a complex and heterogeneous system landscape, (2) the combination of heterogeneous data sources for the purpose of architecture information recovery, (3) the consolidation of available and recovered information in a unified and integrated data model, the digital architecture twin, and (4) the co-evolution of architectural information with the actual system over time, resulting in architecture description suited for the needs of the architect and different stakeholders [12, 2]. We intend to combine existing architecture recovery methods and to automatically integrate the results in a digital twin, thus providing an extensive overview of the system using different views. Moreover, we perform these methods continuously, leading to evolving, version-aware architecture information about the system,

preventing architectural information decay.

3. Proposed solution

The idea of the approach is to automatically create and co-evolve a Digital Architecture Twin (DARt) of heterogeneous and evolving systems. In general, a Digital Twin is a virtual representation of a physical or non-physical object (Physical Twin) or process often used in the digitalization of cars or engines and enabling the exchange of data and information between the Digital Twin and Physical Twin. This allows to effectively simulate situation and adaptations without tinkering with the real world image, which could result in high costs or unfavorable failures [13]. DARt automatically recovers, consolidates and maintains comprehensive architectural information from heterogeneous data sources as an unified architecture information model. Whenever a data source (e.g., source code) changes, the recovered information is updated in the DARt. The co-evolving DARt is extended by integrating versions of a system over time, incorporating the evolution and current status of the system in the DARt.

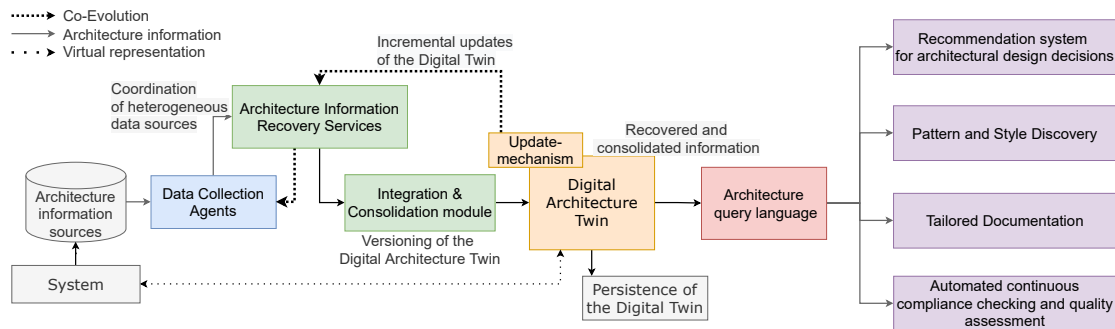


Figure 1: Digital Architecture Twin Generation Process

The Digital Architecture Twin generation process, shown in figure 1, begins with the collection of architectural data from architecture information sources using *Data Collection Agents (DCA)*. Next is the provisioning of this data for the architecture recovery methods implemented as *Architecture Information Recovery Services (AIRS)*. For this, we combine proven architecture information recovery approaches leveraging different data sources. We integrate and consolidate the results provided by the architecture recovery methods into a unified architecture information model based on meta models representing the *Digital Architecture Twin* to obtain an overarching representation of the system architecture. When the system evolves (e.g., source code changes), the DCA and AIRS update the existing architecture information maintaining information of old versions. For this, the update-mechanism triggers the AIRS either automatically or periodically, depending on the source, to retrieve the current architecture information. This updates the DARt incrementally and keeps it up to date with the evolving system. An open challenge is to ensure that the DARt is conform with the system. To use the collected architecture information, an architecture query language will be developed to dynamically retrieve architecture information of different versions, views or abstractions levels of a system and its architecture information.

The DArT in combination with the architecture query language enables to dynamically query information that can be tailored to the specific requirements of developers, architects and other stakeholders at the desired abstraction level and system version. An optional visualization of the dynamic queries and views shall result in human-readable architecture description.

Potential use cases of the DArT are: guided architecture design via architecture recommendation and continuous compliance checking to prevent architecture drift. Guided architecture designs allow based on specific questions and similarity matching of existing architecture information tailored architectural proposals enabling a consolidated IT landscape. Automated and continuous compliance checking monitors whether the actual system has diverged from the planned design (software architecture drift/erosion). Recovered architecture information (as-is) can be compared to the explicitly documented system architecture (as-planned) in order to detect and counteract increased erosion at an early stage.

4. Research method

The process of the doctoral research is displayed in Fig. 2. The first step is a systematic literature review concerned with architecture recovery methods to get a thorough overview of existing approaches, their potential use cases and benefits as well as their limitations. The second step is the identification of potential data sources for the extraction of architecture data, which can be used to recover architecture information employing architecture recovery methods. The third step is the implementation of suitable architecture recovery methods. The fourth step is the creation of the DArT by consolidating the recovered architecture information into an architecture information model built specifically for the integration of static, dynamic and deployment information. The fifth step is the development of an architecture query language built for the retrieval of tailored, stakeholder-dependent architecture information employing the DArT. The sixth step is the conduction of case studies and expert interviews, which are performed iteratively, to evaluate the benefits and understand possible customization of the approach. This evaluation is done using a prototype, which will be developed to extract the architecture and evolution information which enables the generation of the DArT.

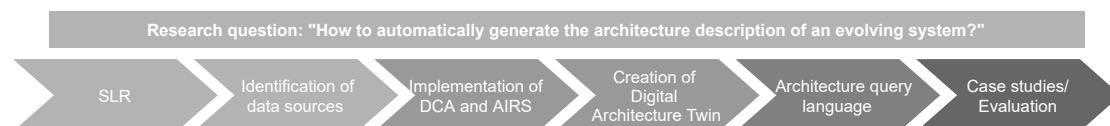


Figure 2: Research method for the dissertation proposal

5. Expected contributions and future work

This work contributes the following: (1) development of the conceptual idea of the DArT for the description of a system, (2) consolidation of available and recovered architecture information in a DArT, (3) co-evolution of the DArT with the system using incremental updates featuring heterogeneous architecture artifacts from different points in time and (4) development of a query

language capable of retrieving architecture information for tailored stakeholder perspectives using the DArT.

We plan to evaluate our approach by developing a prototype of the proposed approach, applying it to open source applications and real-world applications in industry. Furthermore, we intend to perform expert interviews to gather feedback on the idea and approach. Possible limitations of the approach can be the resource- and time intensive architecture recovery process, leading to time shifted description of the analyzed system, devaluing the digital twin. Another limitation is the difficult integration of heterogeneous sources and the consolidation of potentially contradicting information extracted from different sources. Future work comprises of the development of the exchange from DArT to system.

References

- [1] P. Kruchten, The 4+1 view model of architecture, *IEEE Softw.* 12 (1995) 42–50.
- [2] T. Lutellier, D. Chollak, J. Garcia, L. Tan, D. Rayside, N. Medvidovic, R. Kroeger, Comparing software architecture recovery techniques using accurate dependencies, in: *ICSE (2)*, IEEE Computer Society, 2015, pp. 69–78.
- [3] V. Tzerpos, R. C. Holt, ACDC: an algorithm for comprehension-driven clustering, in: *WCRE*, IEEE Computer Society, 2000, pp. 258–267.
- [4] O. Maqbool, H. A. Babri, The weighted combined algorithm: A linkage algorithm for software clustering, in: *CSMR*, IEEE Computer Society, 2004, pp. 15–24.
- [5] P. Andritsos, V. Tzerpos, Information-theoretic software clustering, *IEEE Trans. Software Eng.* 31 (2005) 150–165.
- [6] J. Garcia, D. Popescu, C. Mattmann, N. Medvidovic, Y. Cai, Enhancing architectural recovery using concerns, in: *ASE*, IEEE Computer Society, 2011, pp. 552–555.
- [7] D. Link, P. Behnamghader, R. Moazeni, B. W. Boehm, Recover and RELAX: concern-oriented software architecture recovery for systems development and maintenance, in: *ICSSP*, IEEE / ACM, 2019, pp. 64–73.
- [8] A. Shahbazian, Y. K. Lee, D. M. Le, Y. Brun, N. Medvidovic, Recovering architectural design decisions, in: *ICSA*, IEEE Computer Society, 2018, pp. 95–104.
- [9] D. A. Tamburri, R. Kazman, General methods for software architecture recovery: a potential approach and its evaluation, *Empir. Softw. Eng.* 23 (2018) 1457–1489.
- [10] H. A. Müller, S. R. Tilley, K. Wong, Understanding software systems using reverse engineering technology perspectives from the rigi project, in: *CASCON*, IBM, 1993, pp. 217–226.
- [11] M. S. Laser, N. Medvidovic, D. M. Le, J. Garcia, ARCADE: an extensible workbench for architecture recovery, change, and decay evaluation, in: *ESEC/SIGSOFT FSE*, ACM, 2020, pp. 1546–1550.
- [12] G. Canfora, M. D. Penta, L. Cerulo, Achievements and challenges in software reverse engineering, *Commun. ACM* 54 (2011) 142–151.
- [13] E. Negri, L. Fumagalli, M. Macchi, A review of the roles of digital twin in cps-based production systems, *Procedia Manufacturing* 11 (2017) 939–948.