This paper has been presented on WICSA/CompArch 2016 and will be published by IEEE.

Architectural Data Flow Analysis

Stephan Seifermann
FZI Research Center for Information Technology
Karlsruhe, Germany
seifermann@fzi.de

***Abstract*—Quality properties including performance, security and compliance are crucial for a system's success but are hard to prove, especially for complex systems. Data flow analyses support this but often only consider source code and thereby introduce high costs of repair. Data flow analyses on the architectural design level use call-and-return semantics or event-based communication between components but do not define data flows as first class entities or consider important runtime or deployment configurations. We propose introducing data flows as first class entities on the architectural level. Analyses ensure that systems meet the quality requirements even after changes in e.g. runtime or deployment configurations. Having data flows modeled as first class entities allows analyzing compliance with privacy laws, requirements for external service providers, and throughput requirements in big data scenarios on architectural level. The results allow early, cost-efficient fixing of issues.**

I. Introduction

The complexity of software systems and especially the importance of non-functional requirements including performance, security, or compliance with law increases. Ensuring quality properties is hard and often shifted to the operations phase. Fixing issues found in this phase is, however, costly. Data flow analyses such as JOANA [1] support proving quality properties such as non-inference but often require source code. Related security modeling approaches such as UMLSec [2] require a fine-grained design.

We could not identify an architectural description language (ADL) in the survey of Aleti et al. [3] that explicilty models data flows. Instead, they rely on call-and-return semantics or event-based communication for determining data flows. They do not consider runtime configurations including access control or deployment configurations that affect the data locations. These configurations clearly affect quality properties, especially when talking about privacy concerns.

We propose first class modeling of data flows by extending an existing ADL with data sources, sinks and processing operations. Architects shall model the envisioned data flows and define data related quality requirements as goals derived from user requirements, laws, or service level agreements. We integrate existing deployment and runtime configurations into the analyses to check the fulfillment of requirements even after configuration changes. A constraint solving approach provides the analysis results. A mapping between these results and architectural elements enables comprehensibility.

The data flow analysis enables early detection of requirement violations regarding privacy laws, external service providers, and throughput requirements in big data scenarios on the architectural level. Fixing them is cost-efficient and thereby establishes quality-by-design.

The remainder of this paper is structured as follows: Section II covers the data flow modeling and analysis. In Section III, we discuss the potential impact on industrial practice. Section IV concludes the paper.

II. Architectural Data Flow Analysis

We suggest realizing data flow analyses by establishing data flows as first class entities, transforming the model into constraints, solving them, and mapping results to architectural elements. We suggest extending the ADL PCM [4] because of its extensive architectural simulation capabilities. The following paragraphs cover the modeling and analysis in detail.

We favor integrating data flows into an existing ADL to reuse models and thereby reduce the modeling effort. Figure 1 shows the PCM integration for a simplified version of the MediaStore case study [4]. A *WebGUI* component provides services via the *IWebGUI* interface. The user interface uses a media and a user management that both use databases to store information. Databases are deployed on a server that is subject to another jurisdiction. The *ad Usage* diagram models a data source, data, and the transferring of data for the initial call to the user interface. The *ad Register* diagram shows how the *UserManagement* component processes the data. The note on the lower left corner is an exemplary analysis goal.

The architects model data on a meta-level, which means types of data rather than concrete data are the modeling subject. For instance, architects model that a user provides name and nickname as part of a registration but do not model that a user *John* uses the nickname *johndoe* during the registration. Data and data flows have meta-data such as privacy level, and required bandwith. In case of sensor data, they could have resolution, and age properties. Data sources such as the user in the example from Figure 1 emit data and initialize their meta-data. Data processors affect meta-data and transfer data between components. Data flow specifications such as *ad Register* define the data processing. For instance, a processor encrypts data before transmitting it to the next component and thereby reduces its requirements on privacy. Data sinks are storages or users and terminate data flows. Black boxes with modeled assumptions, e.g. about the server location, represent third-party components such as cloud services. Data processors that use runtime configuration such as access control require separate modeling or a retrieval interface for policies.

The data flow analysis operates on two inputs: The architectural model extended by data flows and potential additional information sources such as access control policies, and the analysis goals. The architect can choose between predefined and self-defined goals. The goal given in Figure 1 could be predefined because it is commonly used in privacy laws. A

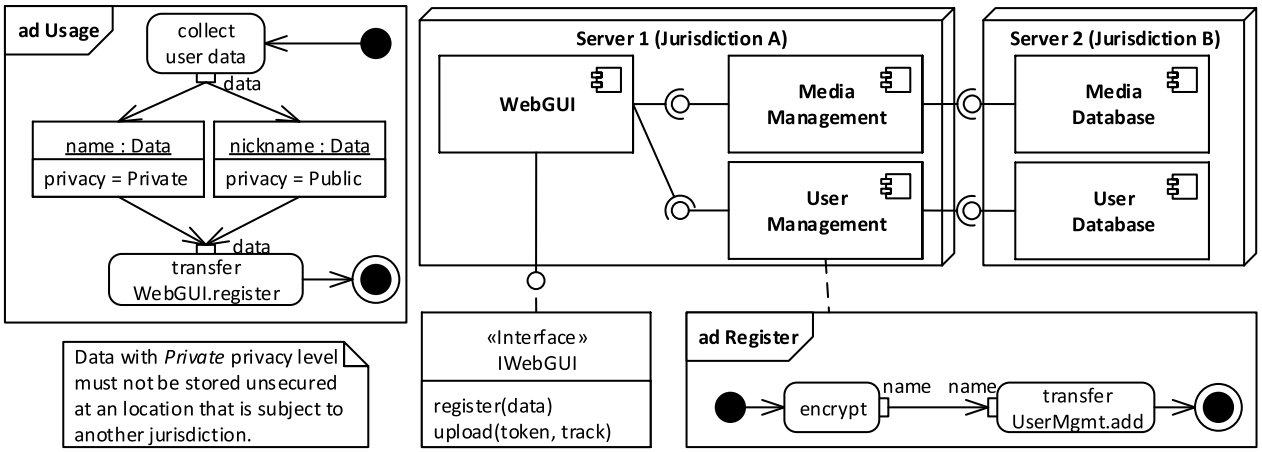

Figure 1. Overview of the envisioned PCM data flow modeling extension (*ad Usage* and *ad Register*) for a simplified web media store.

model transformation uses the inputs to generate constraints. A standard constraint solver determines if the analysis goals can be met for the generated constraints. The choice for a constraint solver depends on the analysis goals. For instance, constraint logic programming is sufficient for privacy related data location analyses while sophisticated access control mechanism might require more powerful constraint languages. Mapping solver results to architectural elements is the last step and important to make results comprehensible for architects.

Goals usually compare normative with derived data properties. For instance, the goal in Figure 1 compares actual with allowed data locations. Such *horizontal* analyses through the architecture determine data properties at any architectural point and check if user or provider requirements are met. *Vertical* analyses check if the infrastructure such as provided by cloud services supports the user requirements for the overall system.

III. Potential Impact on Industrial Practice

We see a potential impact on industrial practice in three areas: Improved development processes, easier negotiation with external service providers, and improved customer experience.

The data flow analysis is another building block for integrating quality requirements in early design phases. This allows detecting issues and fixing them with less costs than required in later phases. Especially for security concerns, this becomes possible because even non-security experts can carry out impact analyses of changes and decide for more secure solutions. Code-based data flow analyses do not allow cost-efficient fixing because the potential bad design decision has to be implemented first. Additionally, these analyses miss runtime and deployment configurations. Our analysis allows finding new security issues related to the operations phase.

Contract negotiation with external service providers becomes easier as contracts can be tailored to the needs found in our analyses exactly. For instance, vendors know the data transferred to an external service and can negotiate server locations to comply with privacy laws that prescribe them.

Using the analysis results can improve user experience: Vendors become confident about data usage and compliance with laws – especially privacy laws. Users can rely on vendor statements regarding privacy and choose services accordingly. Compliance statements are possible timely after changing systems that utilize external services. Users are no quality testers anymore after service changes but the provider determines quality properties before application deployment. This prevents the vendor's image from damage. Especially, the analysis also tackles context and performance related challenges for secure software evolution defined in our previous work [5].

IV. Conclusion

In this paper, we argued for integrating data flows as first class entities into the existing ADL PCM to enable new analyses regarding privacy or SLAs, for instance. The data flow modeling reuses existing architectural descriptions and thereby lowers the preparation and consistency overhead for the analysis model. A constraint solver produces analysis results that are mapped to the architecture for comprehensibility. We also discussed the impact on industrial practice.

We plan to implement a prototype of our modeling and analysis approach for a simple privacy related scenario and later for more complex scenarios. Thereby, we want to investigate if the required data is available at architectural design time and how much effort the extension of existing architectural descriptions requires in practice.

Acknowledgments

This work was supported by the German Federal Ministry of Labour and Social Affairs under grant 01KM141108.

References

[1] G. Snelting *et al.*, "Checking probabilistic noninterference using joana," *it - Information Technology*, vol. 56, pp. 280–287, 2014.

[2] J. Bürger *et al.*, "Restoring Security of Long-Living Systems by Co-Evolution," in *COMPSAC'15*, vol. 2. IEEE, 2015, pp. 153–158.

[3] A. Aleti *et al.*, "Software architecture optimization methods: A systematic literature review," *IEEE Transactions on Software Engineering*, vol. 39, no. 5, pp. 658–683, May 2013.

[4] Reussner, Ralf H. et al., Ed., *Modeling and Simulating Software Architectures – The Palladio Approach*. MIT, 2016, iSBN: 978-0-262-03476-0, to appear.

[5] S. Seifermann *et al.*, "Challenges in secure software evolution - the role of software architecture," in *EMLS'16*, 2016, accepted, to appear. Available at http://sdqweb.ipd.kit.edu/publications/pdfs/seifermann2016b.pdf.