

SERVICE MODELING: THE KEY TO COMPLEXITY MANAGEMENT FOR DISTRIBUTED AND REACTIVE SYSTEMS

– Position Paper –

Ingolf H. Krüger
University of California, San Diego
Computer Science and Engineering Department
9500 Gilman Drive
La Jolla, CA 92093-0114
USA

Applications of interacting software systems have become omnipresent over the past decade. Advances in enabling technologies, such as wireless networking and communication, have brought computer systems into almost every aspect of daily life. Many areas of individual and mass transportation have become almost completely dependent on the correct functioning of software systems. Cars, for instance, rely heavily on the interplay of dozens of software-controlled systems, some of which manage highly safety-relevant parts of functionality. By extending the reach of information technology and infrastructure further into the physical world the need for correctly designing and managing the interactions of large, highly distributed systems will increase even more dramatically. The complexity of these systems, where individual components will contribute only partially to an overall goal, calls for an innovative software engineering approach that places the interaction of components in the center of concern. Such an approach will treat the overall goal, i.e. the *service* delivered by the interplay of individual components, as first-class elements of requirements capture, analysis, design, and implementation.

Contrary to the importance of services and their defining interaction patterns, however, most established, industrially relevant software development approaches and modeling languages, place their focus on the construction of *individual* software components instead of on their interplay.

We conjecture that unless the UML and its cousins adopt services as a central modeling element its corresponding tools will not mature towards true support for systematic software construction, especially in the context of critical systems design.

Therefore, it is proposed to investigate a methodologically founded basis for service-oriented software and systems engineering. This will provide the foundation for systematic development of next-generation systems in which the management and correct implementation of the *interplay* between software components across multiple networking platforms is the central development challenge.

From a software engineering point of view central topics to be addressed in this context are:

- How to capture the key interactions between components in highly distributed and heterogeneous environments? What are the description techniques we provide to the software engineer for representing the central interaction patterns within such systems to gain a better understanding of complex system dynamics?
- How to derive *logical* software architectures from the captured interactions? The purpose of logical software architectures is to serve as a model of the system under development on a high level of abstraction, where specifics of the implementation platform are irrelevant. To what extent can this derivation be automated, giving rise to opportunities for building more powerful tools for software engineering?
- How to transform logical software architectures into *implementation* architectures, taking into account the infrastructure of the system where a service gets deployed. The transition from a

logical model to one that is suitable for implementation is crucial and error-prone if carried out manually. Can systematic steps leading from an abstract model to implementation infrastructures for executing the services under consideration be identified? If so, can these steps also be supported by advanced software engineering tools?

- How to validate and verify the correctness of services based on either the logical or the implementation software architecture? Can a formal representation of a service be devised that supports more powerful and scalable testing and verification techniques as compared to the ones available today?

A concrete research agenda that tackles these challenges might address the following themes in the range from precise methodological foundations to application development.

Foster a solid understanding of services and their properties: A profound mathematical foundation for services is crucial to the systematic development of high-quality systems. It decides how much information is available to reason about the system under development, and how scalable the overall approach is; thus, it has a direct impact on the level of understanding obtainable of interaction- and service-oriented systems. A careful analysis and adaptation of existing mathematical models for system behavior will lead to a precise service notion, integrating both static and dynamic aspects of component interaction.

Make services as modeling-elements available to engineers via expressive description techniques: Enhancing the ability of engineers to construct correct software systems is of paramount importance, given the increased demand for high-quality software systems outlined above. It is proposed to investigate the potentials of new, hierarchic notations focusing on services and their defining interactions; to that end, existing notations – such as Message Sequence Charts and elements of the Unified Modeling Language (UML)– should be investigated, and adapted to the formal service model.

Install the notion of service as the driving metaphor in system design and quality assurance: Thinking in terms of services triggers a fresh look at how software-architectures are crafted; if services – not components – are the driving entities in the initial phases of software design, methodological steps need to be conceived to map services to the components implementing them. In view of the magnitude of the envisioned systems mechanisms for iterative, hierarchic architecture transformations with algorithmic support is essential; prime consideration should be given to establishing both the correctness and practical applicability of these transformations. The interaction-based service approach also delivers an immediate anchor-point for collecting design knowledge, and making it available in the development process. Experimental tool platforms and test-beds will enable a thorough evaluation of the results with respect to other methodologies.

As a result, the notion of service will emerge as a first-class *modeling* element – instead of being only a first-class *implementation* element. Its very definition as a cross-cutting interaction aspect links it directly to implementation techniques such as *aspect-oriented programming*. This bears significant potential for addressing the increasing complexity of distributed, reactive systems across development phases.

Dr. Ingolf Krueger holds the position of Assistant Professor in Residence within the Department of Computer Science and Engineering at UCSD. He holds a Ph.D. from the Technical University of Munich, Germany, and an M.A. from the University of Texas at Austin. Dr. Krueger has co-organized workshops on service-oriented software engineering (FSE'03) and scenarios and state-machines (ICSE 2002, ICSE 2003), which are related to the material presented here. His major research interests are software & systems engineering for distributed, reactive systems, service-oriented software architectures, description techniques, verification&validation, and development processes. Publications pertaining to these areas are available at URL: <http://www-cse.ucsd.edu/~ikrueger/publications.htm>