

## Research Statement

Software engineering is the area in which most of my research is focused, although my research interests also include programming languages. My research is motivated by the need for greater productivity in software engineering while at the same time addressing the increased complexity of a highly wired world. The main goal of my research is *to improve cost-effectiveness in the development of dependable, scalable, and adaptable large-scale software systems*. Current practice usually requires the employment of a significant amount of expensive, repeated labor for achieving these three qualities in software systems. Software architectures have emerged on the scene with a promise to provide such desirable qualities, while checking the costs of developing modern, large-scale software systems. A software architecture allocates system function across its elements, determines the configuration whereby these elements are organized into the system, fixes the nature and protocols of interaction required between these elements, and specifies the data exchanged in these interactions.

The design of software architectures for dependable and adaptable software systems remains an area of active research. Ideally, it should be possible for an architect to put together a recipe for the construction of a software system, relying on past experience and well-known architectural principles, using compositional techniques and notations. One approach to reduce the difficulty of designing and constructing software architectures is aimed at achieving high-level reuse through the use of an architectural style for designing the architecture of a family of similar software systems. Architectural styles are coordinated constraints on similar architectures and embody high-level reuse, support stylistic analysis of architectural properties, and guarantee these across style-based architectures. Therefore, my research has focused on the construction of style-based software architectures.

Practical architectural construction requires support in the following key areas: architectural comprehension, compositional design, behavioral analysis, implementation, and environments. My research has focused on each of these areas simultaneously. The underlying basis of my research is the hypothesis that systematic compositionality of software architecture will lead to improved dependability and adaptability of large-scale software systems.

To support architectural comprehension, I have developed two different but related techniques: one aimed at the classification of software connectors, and another for the characterization of architectural styles. Software connectors have come to be accepted as first class elements of architectures. My comprehensive classification of different software connectors used in software systems and lays the ground for their composition. This taxonomy of connectors is regarded in the software architecture community as the most comprehensive survey of the field to date. I have also developed a novel approach for characterizing architectural styles along five orthogonal dimensions: data, structure, interaction, behavior, and topology. This approach has been successfully applied to more than twenty styles for network-based systems – a major class of modern, large-scale software systems.

The next step of architectural construction requires theories and models for the ground-up composition of architectural elements from fine-grained primitives, which is lacking in the existing literature. This often leads to the lack of compositionality and inconsistency between architec-

tural models and actual systems. To help bridge this gap, my research has exclusively focused on a composition theory and model for constructing software architectures from a small set of fine grained, executable, *architectural primitives*. I have developed an architectural composition framework, called *Alfa*. The Alfa framework has been applied for the composition of network-based software systems. Moreover, I have developed a comprehensive, effective, and automated approach for checking the conformance of software architectures to the style(s) used in them. This assures the presence of desirable stylistic properties in the architectures built using that style.

One facet of my research is the ability to effectively model check software architectural compositions for behavioral properties such as liveness and safety. My research has developed effective techniques for modeling architectural assemblies for determining such properties. Behavioral models are created using a notation for constraint automata called CoLa that I have developed at USC. These models are capable of dealing with architectural assemblies consisting of hundreds of primitives, and effort is currently underway to scale these models to thousands of primitives.

An important distinction of our work from related research is the support for architectural implementation. I have co-developed architectural middleware support for the implementation of software architectures. Work is also currently underway to provide such support for the implementation of Alfa's architectural compositions.

Last but not the least, comes the need to provide effective environments for modeling architectural compositions. We have, therefore, developed a visual environment for modeling Alfa style-based architectural compositions as a domain-specific modeling environment, using a visual notation, called xAlfa. Coupled with the style conformance checking of architectures, this provides an integrated environment for constructing style-based software architectures.

My current research is an attempt to pursue my longer-term research goals through an investigation of several issues including:

- Can development cost-effectiveness and runtime scalability be gained by systematically performing step-wise refinement and optimizations of architectural abstractions?
- How can the analysis of structural and behavioral properties of architectural assemblies be scaled to tens of thousands of primitives such that the analysis can be performed on a typical work station?
- How can a large class of software connectors be composed from architectural primitives? Can such a composition technique help reduce the complexity of adapting the software connectors for requirements evolution?
- How can optimization techniques such as those developed in the field of programming language design be applied to architectural assembly languages to produce correct and scalable architectural implementations?
- How can an industry standard such as UML 2.0 aid in cost-effectively modeling and implementing software architectures?