

Future Directions in Software Design Research

Martin Rinard
MIT CSAIL
Cambridge, MA 02139

Designs are important because they look both ways: they provide the connection between our ambitions (as expressed in the requirements and goals of our envisioned systems) and our realization of those ambitions (in the form of an acceptable implementation). Several research directions hold out the promise of substantially increasing the utility of design for software engineers, with corresponding benefits for our ability to engineer successful software systems:

- **Connections Between Design Models:** Different design models focus on different aspects of the system — some, for example, focus on the structure of its state, while others focus on the structures that realize its behavior. There is a need for new techniques that help developers understand how these different design models combine to interact in the implemented system.
- **Relationship Between Design and Implementation:** One important purpose of the design is to guide the implementation of the system. There is a need to establish a precise connection between the design and the system and to develop new analysis techniques (both static and dynamic) that use the connection to verify that the system correctly implements its design.
- **Relationship Between Design and Requirements:** Another important purpose of the design is to structure the realization of the requirements in an implemented system. Just as there is a need to better relate the design to the implementation, there is also a need for new techniques to improve the connection between the design and the requirements. These techniques might include new languages (both design languages and requirements languages) and analyses that enable us to better explore the relationship between the requirements and the design.

1 Connections Between Design Models

Different design models focus on different aspects of the design. This is useful and appropriate because one of the purposes of design is to identify the key elements of the envisioned system, and different elements are relevant for different design and implementation activities.

Eventually, however, it becomes necessary to understand how all the different elements will come together in the final system. To support this activity, it would be useful to develop families of design models that interoperate with each other, linguistic mechanisms to express important concerns that arise when considering properties of the combined models, and new analyses and simulations that help explore the structure and behavior of the combined design.

The overall goal is to better understand the *span* of the combined design — the range of structures and behaviors that it identifies. Such an understanding would help us reason about the design to understand if the systems that we build will interoperate correctly with other systems and have the desired effect when deployed. They will also help us to more effectively explore the design space.

In this context, an important issue is expanding the range of design elements that one can bring into the combined models. The core of current models focuses on the construction of entity-relationship models and closely related phenomena. Future design languages should also include support for other important, widespread elements of the design such as the intended real-time behavior.

2 Relationship Between Design and Implementation

It is important for the system to correctly implement its design — otherwise, the system may exhibit undesirable behavior and the design may lose much of its value as a tool for reasoning about the system. Consider that even though much of the design of a physical system is manifest in its realization, complex physical systems often use mechanisms such as color coding, status indicators, and packaging to make the relationship between the design and the system (specifically, its structure and behavior) more apparent.

A key challenge facing all stakeholders of software systems is the need to better understand the structure, behavior, and consequences of using the system. Because of recent advances in areas such as design formalisms, programming languages, software engineering, and program analysis, we have an opportunity to establish meaningful, guaranteed connections between the software system and all aspects of its design. These connections will provide a host of benefits: they will improve our ability to understand the system and use it productively in a variety of contexts (and enable us to avoid using the system in inappropriate contexts!), help us to identify and address inadequate or incomplete designs, increase the reliability and security of the system, help us to detect potential software development quagmires much earlier, and help structure the development process to reduce development time and cost.

The way to obtain these benefits is to develop explicit, checked connections between the design and its implementation. These connections will specify the relationship between the implementation (its code, data structures, and behavior) and the design. Examples of such connections might include abstraction functions that specify how efficient, heavily encoded data structures map to concepts in the problem domain or how the code is decomposed into subsystems. These connections, in turn, can help designers verify important properties of how the different aspects of the design interact.

After development, we expect that it should be possible to take the design information embedded in the implementation, then automatically extract design elements. Among other benefits, this connection will ensure that the design continues to faithfully reflect the implementation as the implementation is modified during maintenance.

3 Relationship Between Design and Requirements

The goal of most design activities is to help bridge the gap between the implementation and the requirements. Just as the connection between the design and the implementation is important, so is the connection between the design and the requirements — one goal of the design is to (partially) identify properties of a system that realizes the requirements. We therefore need ways to relate the design to the requirements. Design exploration via simulation followed up by developer interpretation of the results is one informal way to perform this activity.

It is time to investigate a more systematic approach. There needs to be a way for the developer to identify key properties that the system should satisfy (each of these properties formalizes a requirement), then reason about the design to see if it is likely to satisfy these properties. The key research issues include finding a good language in which to express the requirements and developing new analyses and simulations that relate the design to the requirements.