

A Refinement-Based Framework for a Science of Design

Douglas R. Smith
Kestrel Institute
Palo Alto, CA 94304
smith@kestrel.edu

Software can be seen as a composition of many sources of knowledge, including requirements, system and programming knowledge, the target programming language and platform, as well as legacy components. We look to a design of science to provide a practical but well-founded mathematical basis for mechanizing the design processes in software-intensive systems. We advocate an approach that works with

1. *formal specifications of software requirements* – the design process is driven by the goal of correctly implementing the requirements. The larger problem of evolution is likewise driven by modification of the requirements specification followed by redesign.
2. *libraries of design theories* that capture important recurring design concepts. Examples include generic software architectures (capturing knowledge about the structure and design of various classes of systems), algorithm theories (capturing knowledge about the structure and design of various classes of algorithms), datatype refinements (capturing knowledge about abstract datatypes and effective concrete implementation types for them), optimization transformations, and so on.

The overall development process involves an iterative convergence towards a requirement specification that suits the needs at hand together with composition of the programming knowledge needed to transform the evolving specification to correct-by-construction code. The development of a specification uses composition of theories to enable reasoning about the application domain. The application of formalized representations of design knowledge requires inference to determine the applicability of abstract knowledge to the concrete application at hand, and the composition of the instantiated knowledge to the application.

In the particular approach that we have implemented at Kestrel in the Specware system, we work in one of several categories of specifications that support composition by colimit, and refinement of specifications by morphisms (interpretations between theories).

Besides rationalizing the design process and drawing on tools from many disciplines (e.g. theorem-proving, program analysis, databases of specification, design, and program structure, libraries of theories, libraries of components), a refinement-based compositional approach to design moves in the direction of supporting the kinds of flexibility/variability needed to accommodate evolution. Additionally, the information needed to rationally support software evolution (and provide machine support) is naturally provided by a formal generation process. That is, evolution is driven by changes in requirements, manifesting in changes to required properties (necessitating redesign of the code), or changes in performance requirements (necessitating change in the design knowledge applied), or changes in nonfunctional requirements such as robustness (e.g. necessitating introduction of more fault-tolerant architecture). By having an explicit handle on the locus of change, we have a better basis for a rational design process.

Background

Dr. Douglas R. Smith has been with the Kestrel Institute since 1984. He is also Executive VP and CTO of Kestrel Technology LLC. He taught an advanced graduate course on knowledge-based software development at Stanford University during 1986-2000. He is a Fellow of the American Association of Artificial Intelligence (AAAI) and served as Chairman of IFIP Working Group 2.1 on Algorithmic Languages and Calculi from 1994-2000.

Dr. Smith's main research interest has been mechanizing the development of software from formal specifications. He is mainly responsible for the development of KIDS (Kestrel Interactive Development System), which has been used to generate many algorithms, including a variety of complex high-performance scheduling applications for the US Air Force. Current projects are focused on the development of the Specware-based systems which provide category-theoretic foundations for software development by refinement. Recent systems include Designware (general-purpose refinement), Planware (synthesis of schedulers and resource allocators), and Epoxi (specification and refinement of hybrid embedded systems).

Dr. Smith has over 25 years experience in the field of automated program generation and has published over 70 papers.

Selected Publications

1. Marcel Becker, Limei Gilham, Douglas R. Smith, Planware II: Synthesis of Schedulers for Complex Resource Systems, submitted for publication, 2003.
2. M. Burstein, D. McDermott, D.R. Smith, and S.J. Westfold, Derivation of Glue Code for Agent Interoperation, invited paper in *Journal of Autonomous Agents and Multi-Agent Systems* 6, 2003, 265-286.
3. Dusko Pavlovic and Douglas R. Smith, Composition and Refinement of Behavioral Specifications, *Proceedings of the Sixteenth International Conference on Automated Software Engineering*, IEEE Computer Society Press, Coronado Island, CA, 2001, 157-165.
4. Smith, D.R., Mechanizing the Development of Software, in *Calculational System Design, Proceedings of the NATO Advanced Study Institute*, Eds. M. Broy and R. Steinbrueggen, IOS Press, Amsterdam, 1999, 251-292.
5. Smith, D.R., Toward a Classification Approach to Design, invited paper in *Proceedings of the Fifth International Conference on Algebraic Methodology and Software Technology*, AMAST'96, LNCS 1101, Springer Verlag, 1996, 62--84.
6. Smith, D.R., Constructing Specification Morphisms, *Journal of Symbolic Computation*, Special Issue on Automatic Programming, Vol 16, No 5-6, 1993, 571-606.
7. Smith, D.R., KIDS: A Semi-Automated Program Development System, *IEEE Transactions on Software Engineering* 16(9), Special Issue on Formal Methods, September 1990, 1024-1043.
8. Smith, D.R. and Lowry, M.R., Algorithm Theories and Design Tactics, *Science of Computer Programming* 14(2-3), Special Issue on the Mathematics of Program Construction, October 1990, 305-321.