Determining the Cause Design Model Inconsistencies

John Grundy, *Swinburne University of Technology*

This installment of *Computer*'s series highlighting the work published in IEEE Computer Society journals comes from *IEEE Transactions on Software Engineering*.

Don't you just hate it when you find an error in your design and then have to work out where it came from? What caused it? Is it a minor modeling error? Or more fundamentally, does it result from an inconsistent design or even a requirements error? These issues have bedeviled designers and programmers alike since the days of the ENIAC. With today's increasingly complex designs and composite systems, identifying and correcting design errors early and accurately is more critical than ever.

In a recent article in *IEEE Transactions on Software Engineering* (vol. 39, no. 11, 2013, pp. 1531–1548), Alexander Reder and Alexander Egyed from Johannes Kepler University, Linz, Austria, bravely tackle this problem. They've produced a major advance in inconsistency analysis—moving beyond just identifying a design inconsistency to accurately showing designers where the inconsistency originates.

Their work starts from the premise that visualizing an inconsistency in a design tool ought to also involve visualizing its cause, a novel concept not previously explored in software engineering design. Understanding the cause is vital in formulating a suitable repair, whether manual or automated. And to understand the cause, it must be understood that behind an inconsistency is a design rule—a constraint—that the design violated. Inconsistency is defined in how the various parts of the design rule contributed to the inconsistency and identifying which model element in the design influenced those parts. To do so, the authors present an equally novel approach for computing the cause of design model inconsistencies by looking at inconsistencies from two angles: the syntactic structure of design rules to understand the expected results of their parts and the validation results to determine where the validated results match the expected results. The cause of an inconsistency is then the set of model elements that contributed to validations that didn't match expectation.

The authors rigorously evaluate this novel technique and supporting prototype tool plug-in using a correctness measure (based on causes being complete and minimal), an effectiveness measure (based on scalability and determining the full set of causes for an inconsistency), and a performance measure (based on computational scalability). They show that their technique can determine the full set of expressions that cause model consistency problems both accurately and scalably.

Given the prevalence of models well beyond software engineering, anyone interested in modeling tools, model inconsistency checking and resolution, and holistic and effective evaluation of engineering research should read this paper.

John Grundy *is dean of the School of Software and Electrical Engineering, Swinburne University of Technology. Contact him at jgrundy@swin.edu.au.*

For more news on IEEE CS transactions, sign up for the newsletter at www.computer.org/newsletters.

Published by the IEEE Computer Society