Award Details

Automated formal methods for reliable software components

Research Details

Competition Year:	2007	**Fiscal Year:**	2011-2012
Project Lead Name:	Beyer, Dirk	**Institution:**	Simon Fraser University
Department:	Computing Science, School of	**Province:**	British Columbia
Award Amount:	15,610	**Installment:**	5 - 5
Program:	Discovery Grants Program - Individual	**Selection Committee:**	Computing and Information Sciences - A
Research Subject:	Software engineering	**Area of Application:**	Computer software
Co-Researchers:	No Co-Researcher	**Partners:**	No Partners

Award Summary

In my research program I work on software verification, which is a method that takes as input a program and gives the user feedback about possible bugs in the program. We already use such techniques today, in our compilers: they check if we use all variables and functions properly, according to their types and signatures, respectively. As a next step towards the ultimate goal of error-free programs, we develop the technology needed for checking semantical properties, e.g., if data structures on the heap are consistent, or if some forbidden variable value is possible during program execution. Theoretically, these checks are undecidable, but in practice, due to several recent breakthroughs in program verification, there is a strong hope that we can actually verify many interesting properties. The problem is identified and widely acknowledged as a grand challenge in computer science, and only an enormous effort of many research groups together can develop all the techniques that at the end make it possible to achieve the ideal of software engineering: verified software. However, it is not always possible to obtain the source code for all components of our system. This phenomenon naturally occurs if we build information systems based on web services. Different components, developed by different development teams, are running at different locations and on different computer platforms. Neither the source code nor the executable program is obtainable for inspection or in-house testing of certain components. In the proposed research, we are working on a new formalism, called web service interfaces, to formulate specifications of web services in a concise way, and develop a tool platform that supports modeling and analysis of web services based on our language. A provider of a web service can then offer to the clients a formal interface of its service. The provider can use the interface to check if the actual implementation fulfills all promises made by the interface. The client of the web service can use the provided interface to check if the client component is compatible with the interface. If both is given, then the overall composition is guaranteed to work. We will provide automatic procedures for both tasks.