Christian-Albrechts-Universität zu Kiel, 24098 Kiel

Prof. RNDr. Jan Kratochvil, CSc.
Dean
Faculty of Mathematics and Physics
Charles University in Prague
Ke Karlovu 3,
121 16 Praha 2
Tschechien

Institut für Informatik
Software Engineering

Leitung:
Prof. Dr. Wilhelm Hasselbring

Hausanschrift:
Christian-Albrechts-Platz 4, 24118 Kiel

Postanschrift: 24098 Kiel

http://se.informatik.uni-kiel.de

Paketanschrift:
Christian-Albrechts-Platz 4, 24118 Kiel

Bearbeiter/in, Zeichen

Mail, Telefon, Fax
hasselbring@email.uni-kiel.de
tel +49(0)431-880-4664 / 3734 (Sek.)
fax +49(0)431-880-7617

Datum
Kiel, 18.07.2014

Review on the doctoral thesis "Instrumentation and Evaluation for Dynamic Program Analysis" by RNDr. Lukas Marek

Dear Prof. Kratochvil,

Concerning the submitted thesis "Instrumentation and Evaluation for Dynamic Program Analysis" I would like to comment on the following points:

- Relevance: static program analysis takes the program's source code to extract information about the program. Dynamic program analysis complements static analysis via observing the program's execution. Dynamic analysis, or the analysis of data gathered from a running program, has the potential to provide an *accurate* picture of a software system because it exposes the system's actual behavior. This picture can range from class-level details up to high-level architectural views. Among the benefits over static analysis are the availability of runtime information and, in the context of object-oriented software, the exposure of object identities and the actual resolution of late binding. This way, dynamic program analysis provides highly valuable information for program comprehension. To enable dynamic analysis, it is required to instrument the program with probes.

  This submitted thesis addresses a highly relevant field in Software Engineering.

- Contribution: Several design decisions are required when instrumenting programs for dynamic analysis.[1] This thesis makes contributions to two such design decisions: How to instrument the program and how to separate the program's execution from the program's analysis in order to reduce overhead and perturbation caused by the analysis. For instrumentation, the author invented DiSL, a domain-specific language for Java byte code instrumentation. The key concepts of DiSL are small runtime overhead, simple extensibility, and observation coverage of the whole Java Class Library. For analysis evaluation, the author invented ShadowVM, a framework for offloading

[1] See also: van Hoorn, A., Rohr, M. und Hasselbring, W. (2009) Engineering and Continuously Operating Self-Adaptive Software Systems: Required Design Decisions. In: Design for Future, October, 2009, Karlsruhe, Germany.

analysis evaluation out of the context of the observed program. ShadowVM uses two virtual machines to prevent perturbation of the observed application. One JVM (the observed VM) is running the native agent responsible for marshaling events from the observed application, while a second JVM (ShadowVM) is performing the actual evaluation.

This submitted thesis provides new innovative contributions to the field of dynamic program analysis.

- Evaluation of the contribution: For a thesis in Software Engineering, I expect an experimental / empirical evaluation of the original contribution.

  The author of this thesis largely contributed to the design of the DiSL language and was also the lead developer and one of the two main authors of the DiSL framework. He also compared DiSL to the popular tools ASM and AspectJ. The author also designed and implemented most of the ShadowVM framework. In my view, this is a good evaluation for this engineering thesis.

  The contained papers were published at the highly specialized conferences AOSD and GPCE, and one paper has been accepted for publication in the established Elsevier journal Science of Computer Programming. Thus, the respective research community also accepted the major contributions of this thesis.

- In Chapter 6, I missed some related work. For the defense, I suggest to let the author compare his contributions to the following instrumentation / monitoring approaches:

  - SPASS-meter: H. Eichelberger and K. Schmid. Flexible resource monitoring of Java programs. Journal of Systems and Software 93 (July 2014), pages 163–186.
  - Kieker: A. van Hoorn, J. Waller, and W. Hasselbring. Kieker: A framework for application performance monitoring and dynamic software analysis. In: Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering (ICPE 2012). ACM, Apr. 2012, pages 247–248.

  Please note that I do not consider it a critical issue for the acceptance of the thesis that these related works are not discussed. However, I suggest to address that in the defense.

In summary, I can confirm that the thesis proves the author’s ability to do creative scientific work.

I propose to the Faculty of Mathematics and Physics at Charles University in Prague to accept this thesis.

Yours sincerely,

Prof. Dr. Wilhelm Hasselbring