Seminar Configurable Systems

Prof. Zeller / Christian Lindig — Project Proposal

Assignment

We have read four papers by now that have introduced important concepts as well as their implementation as tools.

1. Write a one-page proposal for a small project related to software build tools. The proposal should address three points:

   (a) An interesting question to investigate.

   (b) Why you find the question is interesting.

   (c) A plan or experiment how to answer the question. All proposals should have a practical component unless your question is theoretical and you are going to use formal methods for answering it.

   You can work on your proposal in groups of two and submit a single proposal together. Just because you think a certain question is interesting does not mean that you have to answer it. The goal is to find interesting questions. Each of you will present his proposal next week and we will discuss them. I will make copies of all proposals and distribute them.

2. Send me your proposal as a PDF file by email.

Possible Topics

Here are some ideas about possible topics for proposals.

- A proposal does not have to pose a completely new question. For example, the verification of a result from a paper that we have read would make a good proposal. Typical candidates for these kind of questions are claims related to the performance of systems.

- You could try out a system and compare your experience with the claims from a paper. Here it is important to not just try what the paper tried but to explore the limits of the system: is the system really language independent, does it scale to large systems, is its specification language concise?

- There are more software build tools than we have read about and maybe you already have been working with one one them. You could compare it on a conceptual and practical level with tools that we have discussed.

- We talked a lot about the concepts of tools, but none of the papers looked at the projects that get built using these tools. You could investigate and compare several open-source projects from the perspective of building: what does the dependency tree really look like? How much parallel building is possible? How does Make compare on these projects to other tools?
- Make is still the predominant tool for building software. How do users of Make overcome its shortcomings? Are there Make Patterns like there are Design Patterns in object-oriented systems?
- Some tools provide high-level concepts and emit code for Make that possibly is shipped without the high-level specification. How do these tools compare with the tools we have read about?
- If you have a radical idea for software building, you may want to present it and devise a small experiment to validate the core of your idea.