Cost-Effective Evolution of Research Prototypes into End-User Tools: The MACH case study

Harald Störrle[1]

Abstract: Much of Software Engineering research hinges on an implementation as a proof-of-concept. The resulting tools are often quite prototypical, to the degree of having little practical benefit. The Model Analyzer/Checker (MACH) is a case study in turning a set of research prototypes for analyzing UML models into a tool that can be used in research and teaching, by a broad audience. We document how the requirements and constraints of an academic environment influence design decisions in software tool development. We argue that our approach while perhaps unconventional, serves its purpose with a remarkable cost-benefit ratio. This paper is a summary of [St15].

Background For the purposes of research and publication, there is little value in polishing the usability, stability, portability, extensibility, or performance of a tool, as there is only ever one user of the tool. However, this excludes independent replication, benchmarking, human-factors studies (e.g., observing real users actually using the tool), tool usage in teaching settings, or commercial dissemination. Turning a research prototype into a "proper" tool, however, implies substantial engineering effort with little contribution to the research being conducted. Delegating the task to a student as a thesis project may not be possible, or may prove not successful. All too often, the researcher ends up abandoning the further dissemination of a strand of research for lack of resources.

Objective In this particular case, the author's research work focuses on advanced operations on UML models that are beyond the scope of existing modeling tools. Over the years, many small exploratory prototypes have been created, each of which requires a high degree of expertise to use, and presents only a negligible contribution to a modeler. Thus, the objective of MACH was to create a single integrated tool from a set of prototypes to realize synergies, encapsulate it with a user interface to make it accessible to a wide range of users, and, most of all, do all this with as little cost as possible.

Method We focus on academic stakeholders, thus justifying the assumptions that (1) MACH users have some understanding of models and the underlying concepts, (2) they have sufficient motivation to use the tool even if its user interface is less polished than commercial end user software.

Driven by the main rationale of rapid prototyping, most of the prototypes mentioned above have been implemented using the PROLOG programming language. Long-term usage of the resulting code, usage by third parties, or long-term-evolution of the code base was not considered at the time of creation. Thus, creating MACH tried to achieve three goals: (1) Combine most or all of the existing prototypes into a single tool; (2) make the major functions available to students and colleagues; but (3) strictly limit the effort in creating

[1] Department of Applied Mathematics and Computer Science,Technical University of Denmark,`hsto@dtu.dk`

the tool to the bare minimum. We mapped these goals into ten requirements, designed a solution, and evaluated the tool repeatedly in various classes taught by the author.

Result MACH is implemented in PROLOG and provides a command-line user interface, both of which are relatively exotic choices, today. MACH can be obtained online at `www.compute.dtu.dk/~hsto`. There is an online demonstration of MACH 0.93 available via the SHARE platform at [St14], including samples and a manual. No installation is required. MACH uses a set of utility functions and a common file format glued together the following advanced analysis and checking procedures on (UML) models.

- **Clone Detection** important task in the quality assurance of models, where we often find duplicate model fragments, arising through thoughtless "copy-paste modeling".
- **Model Version Control** is as effective as the weakest link in the chain, typically the way differences are presented to human modelers. We have proposed a novel approach and tested it in controlled experiments, but lacked in vivo user studies to provide more reliable evidence.
- **Model size and similarity metrics** are necessary to assess the size and nature of a model. It is straightforward to use the frequency distribution of model element types, and visualize it as a histogram.

In addition to the features described above, MACH provides a number of supportive functions that are essential for doing practical work, such as navigating the directory tree, loading models, handling aliases, command history, file name completion, a help system, and so on. Usage experience so far suggests that our approach serves its purpose.

Conclusions By sticking to an "exotic" high-level language like PROLOG, offering only a simple text interface with ASCII-graphics, and abandoning the idea of a tight, high-productivity integration into an existing tool framework like Eclipse RCP, we can achieve the core goals of deploying research results quickly to a broader audience, while keeping the required effort at the absolute minimum. The approach may be unconventional, but it offers a unique cost-benefit ratio.

We believe this is a viable path for other researchers faced with similar challenges. We hope that our experience will help others make their research prototypes available to larger audiences with reasonable effort, for the benefit of the entire scientific community.

References

[St14] Störrle, Harald: , MACH 0.93. online, 2014. `http://is.ieis.tue.nl/staff/pvgorp/share/?page=ConfigureNewSession&vdi=XP-TUe_XP_SWIPL.vdi`.

[St15] Störrle, Harald: Cost-Effective Evolution of Research Prototypes into End-User Tools: The MACH case study. Science of Computer Programming, 2015. in print.