

USING DESIGN INFORMATION AT RUN TIME

Position Paper for NSF Workshop on
Science Of Design: Software-Intensive Systems

Charles Rich

Mitsubishi Electric Research Laboratories
Cambridge, Massachusetts, USA
rich@merl.com

October 1, 2003

The theme I would like to explore at the workshop is how information about the design of a software system can be used at `_run_time_` to improve the system's usability, reliability, and adaptability. From a philosophical point of view, I see this as an example of "reflection", i.e., we are talking about an architecture in which a system includes a (possibly partial) model of itself. More anthropomorphically, one could also call such systems "self-aware". In general, whether in artificial intelligence or biology, systems with reflection or self-awareness are considered to be on a higher functional plateau than systems without.

To ground these abstractions in a simple example, consider a small, interactive FTP program with a graphical user interface. Normally, the run-time state of such a program includes only the name of the host, the file being transferred, the transfer mode, etc. I think we would all agree, however, that during the creation of the FTP program, the designer possessed, either explicitly or implicitly, a much richer information structure, which included an understanding of the task for which the program was expected to be used, and how the operations in that task map onto particular graphical elements (buttons, check boxes, etc.) in the interface of this program. If this design information was available at run time, the FTP program could, for example, give a novice user a demonstration/tutorial on how to use it. In fact, we have implemented a prototype tool framework which does exactly this, and used an FTP program as the illustration [1].

Notice that the system in this example uses design information at run time, but not to `_modify_` itself---only to explain itself. Design information can, of course, also be very useful for modifying (adapting) a system at run time, such as automatically integrating new components, adjusting to changes in the computing environment, etc. Using design information in this way contributes to the goal of what has recently come to be called "self-healing" or "autonomic" software [2]. In fact, in such a future vision, the distinction between "design time" and "run time" may become obsolete.

One of the key challenges, I believe, in undertaking a successful research program along these lines, is how to choose appropriate "laboratory models". Clearly, the problems this workshop addresses are the problems of very large software-intensive systems. Unfortunately, doing research with such large systems often makes true innovation impossible. At the other end of the spectrum, the FTP program example

above is much too small to be of serious interest. Finding the right middle ground in size, and an advantageous application domain, will be the first order of business.

My background is quite relevant. I was co-PI of the Programmer's Apprentice project [3] at MIT, in which we explored how to capture and use design knowledge during the software development process. In a sense, the theme discussed here is a logical extension of this work to the run time. More recently, I have been working on human-computer collaboration [4].

If there is going to be a "break out" group at the workshop on the theme I have discussed here, or on the related themes of autonomic or self-healing systems, I would especially like to participate and/or help to lead.

REFERENCES

- [1] Eisenstein, J. and Rich, C. "Agents and GUI's from Task Models." ACM Int. Conf. on Intelligent User Interfaces, January 2002, pps 47-54.
- [2] <http://www.research.ibm.com/autonomic>
- [3] C. Rich and R.C. Waters. The Programmer's Apprentice. Addison-Wesley, Reading, MA and ACM Press, Baltimore, MD, 1990.
- [4] C. Rich, C. Sidner, and N.Lesh. "Collagen: Applying collaborative discourse theory to human-computer interaction." AI Magazine, Winter 2001, Vol 22, Issue 4, pps 15-25.