Panel

On the Future of Generators

Moderator: Ira Baxter (Semantic Designs)
Panel Members: James Neighbors (Bayfront Technologies).
Other members to be determined.

Code libraries are the most common form of reuse attempted in organizations. Components are selected by various means and are then composed by informal but simple manual methods, such as text insertion, which leave the components essentially unchanged in the resulting system. This form of reuse has not always been effective or easy to use. For instance, there is no guarantee that a selected component fits into a target system.

An alternative model of reuse has a pre-defined generator mechanism which automatically selects and composes carefully crafted compatible components. Often, the generated code gives little hint about what components were selected. These generators are often effective in producing interesting applications in specific problem domains (Batory: databases, Kant: partial differential equation solvers, Neighbors: communication protocols, etc.). These generators often differ greatly in how composition occurs, how components are represented, and how one describes to the generator the desired result.

This panel will explore the need for generators, and try to compare their goals and mechanisms in order to give the community a better understanding of alternative reuse strategies. The panel members are architects of various generators or conventional reuse methodologies, and will be asked to sketch their generators and compare to other panelists.

Panelists will choose to discuss a subset of the following questions:

Motivation for Generators:
- Are libraries sufficient? For what kinds of tasks?
- What is a "generator"?
- What software engineering task motivated your method?
- Why is a generator needed for your type of task?

General questions:
- How is the problem to be solved described?
- How does your generator choose components?
- How does it compose components?
- Why is that means of composition effective? Is it general?
- How does it affect the way a component is defined? captured? represented?
- Is your tool based on a model/methodology for domain engineering?
- What are key organizing principles behind your tool?
- What issues are poorly addressed by your tool?
- What does a user really have to know to use your tool?

Any universal properties of generators?
- Are there common concepts among generators?
- Are they all domain specific?
- What properties of a domain force a particular style of generator?
- Is there a technological underpinning common to all?

The Future: Pessimistic or Optimistic?
- What are the technical problems ahead?
- What are the non-technical problems?
- Will generators displace conventional reuse? Why/why not?