Tue 4:30-5:15 pm - Rose Ballroom B

Better Science Through Art

- **Richard Gabriel**, IBM Research, United States
- **Kevin Sullivan**, University of Virginia, United States

How do artists and scientists work? The same.

Wed 1:30-3:00 pm - Rose Ballroom B

Rubber Ducks, Nightmares, and Unsaturated Predicates: Proto-Scientific Schemata are Good for Agile

- **Jenny Quillien**, New Mexico Highlands University, United States
- **Dave West**, New Mexico Highlands University, United States

Fine-grain case studies of scientific inquiry, lessons from linguistics on metaphoric thinking, the epistemology of Charles Sanders Peirce, recent work on architectural image-schemata, along with the computer world's own theorist, Peter Naur, all suggest that software developers (frequently dulled and desiccated from overdosing on 'Cartesian' methodologies) could benefit from imbibing a little mysticism—not the wave-your-hands woo-woo kind but the more ineffable hunch and gut side of human cognition. Scholarly publications in their final polished forms rarely admit that stories, jokes, eroticism, and dreams were the fertile seeds that germinated into 'serious' results. This essay looks to these 'closet' sources, non-reductionist, non-self conscious, metaphorical, and aformal modes of thought as the salvation of a profession gone awry. It is notably proto-scientific image-schemata that retain our attention as a pragmatic tool for improving the fecundity of Agile methodology, at its roots, so to speak. The necessary context is provided by Peter Naur's fundamental insights about software development as 'theory building' coupled with an elaboration of the Agile concept of storytelling.

The discussion starts with and, for reasons of length, mainly stays with architecture's Christopher Alexander who offers novel usages of image-schemata and whose older foundational work will be at least somewhat familiar. The reasoning laid out in the essay, however, is general enough to allow the reader to experiment with proto-scientific clues from any source, not just Alexander.

Pure and Declarative Syntax Definition: Paradise Lost and Regained

- **Lennart C. L. Kats**, Delft University of Technology, Netherlands
- **Eelco Visser**, Delft University of Technology, Netherlands
- **Guido Wachsmuth**, Delft University of Technology, Netherlands

Syntax definitions are pervasive in modern software systems, and serve as the basis for language processing tools like parsers and compilers. Mainstream parser generators pose restrictions on syntax definitions that follow from their implementation algorithm. They hamper evolution, maintainability, and compositionality of syntax definitions. The pureness and declarativity of syntax definitions is lost. We analyze how these problems arise for different aspects of syntax definitions, discuss their consequences for language engineers, and show how the pure and declarative nature of syntax definitions can be regained.

Thu 1:30-3:00 pm - Rose Ballroom B

Faith, Hope, and Love - A criticism of current practice in programming language research

- **Stefan Hanenberg**, University of Duisburg-Essen, Germany

Research in the area of programming languages has different facets — from formal reasoning about new programming language constructs (such as type soundness proofs for new type systems) over inventions of new abstractions, up to performance measurements of virtual machines. A closer look into the underlying research methods reveals a distressing characteristic of programming language research: developers, which are the main audience for new language constructs, are hardly considered in the research process. As a consequence, it is simply not possible to state whether an artifact that requires some kind of interaction with the developer has any positive impact on the construction of software. This paper argues for the urgent need for empirical methods in programming language research that rely on studies of developers — and argues that the introduction of empirical methods not only requires a new understanding of research but also a different view on how to educate computer science students.

The Tower of Babel Did Not Fail

- **Paul Adamczyk**, N/A, United States
- **Munawar Hafiz**, University of Illinois, United States

Fred Brooks' retelling of the biblical story of the Tower of Babel offers many insights into what

makes building software difficult. The difficulty, according to common interpretations, comes from the communication and organizational problems in software development. But the story contains one more important lesson that people tend to miss: one cannot accomplish impossible goals, which programmers are often asked to do. Software engineering, as a discipline, can overcome poor communication; but if we attempt to live up to impossible expectations, we will always fail.