Michael Keeling and Mike Velichansky
Agile 2011 *Insights* - Experience Report Proposal
"Making Metaphors that Matter"

Abstract
The architecture metaphor was proposed in Extreme Programming as a lightweight alternative to more rigorous architecture practices, but the metaphor has been shown to be neither effective for team communication nor suitable for evaluating a system's architecture. This is no surprise as few agile teams have had basic training in architecture practices let alone in how to create metaphors. In our experience this does not invalidate the metaphor, but rather shows that more guidance is needed for teams using metaphors. This paper outlines one team's positive experience using architecture metaphors in the development of a medium-sized, service-oriented, enterprise system. Specific guidelines for creating effective metaphors are presented along with concrete examples.

Why this Topic is Important
The architecture metaphor is presented as a technique for improving communication and planning but its usefulness has been questioned since the practice was first introduced. Studies conducted at Carnegie Mellon and elsewhere have shown that the metaphor obfuscates rather than enhances a team's understanding of a software system's architecture and even misdirects communication among teammates. Though the controversial practice has been downplayed in recent iterations of Extreme Programming, the metaphor persists as a misunderstood and poorly executed software design technique. While a metaphor might be better than ignoring architecture completely, teams need clear guidance and training to properly apply the technique.

Until recently architectural thinking had largely been an afterthought in agile development. Traditional architecture practices can be documentation heavy and are often regarded as a waste when the project changes direction. At the same time, a lack of attention to architecture can also lead to waste in the form of rework due to poor design. Some kind of middle ground is necessary for agile projects to be successful. In our experience, metaphors can walk this middle ground by not only providing lightweight guidance but also a measuring stick for evaluating the design of a system as it naturally evolves during implementation.

The architecture metaphor has proved unhelpful because little guidance is available to teach teams how to properly create and apply metaphors. It is absurd to expect two people, even from the same team, to independently come up with the same metaphor to describe a system, as was asked in some studies. Just like a traditional architecture description, team members need to work together to reach consensus and shared understanding around the metaphors they'll use to describe the software they'll build. We propose that metaphors can be extremely useful for the right teams with the right guidance. The goal of this paper is to provide such guidance based on our experiences and to present concrete examples for other teams to follow.

Proposed Outline
1. ***Introduction –*** This section will present the current use of architecture metaphors in agile software development, in addition to basic architectural principles and how they relate to metaphors.
2. ***What is a "Good" Architecture Metaphor*** – We've proposed that metaphors have been applied poorly so we present our definition of what a "good" metaphor looks like. Most generally, the metaphor is a communication tool, not a design tool, that will help a team enforce design decisions but not reason about promoted or inhibited qualities in a design.

When a teammate mentions a metaphor it should bring to mind a set of design decisions and quality attribute considerations and ultimately help teammates make the right design trade-offs. We'll also talk about the benefits we experienced during our project.

3. ***Guidelines for Creating Metaphors*** – To help other teams better create and apply metaphors in architecture design, a set of guidelines for creating metaphors is presented along with common mistakes we encountered.
   - ***3.1. A Metaphor Represents a Single View***
   - ***3.2. A Metaphor gives clear guidance and sets boundaries for implementation***
   - ***3.3. Start with Design (and create a metaphor based on the design)*** – Start with a drawing of the system and a narrative around that visual; do not start with a metaphor.
   - ***3.4. Avoid "Mixed Metaphors"***
   - ***3.5. Avoid Incommunicable Metaphors*** – Communicability is hampered by complexity and abstraction.
   - ***3.6. Draw on Shared Experiences*** – Common experience (both technical and non-technical) is the bedrock of a great metaphor.
   - ***3.7. Make Time to Make Metaphors*** – On our project, sometimes great metaphors were born in a flash, other times it took several days for a metaphor to emerge.
   - ***3.8. Architectural Styles are Metaphors too*** – Don't ignore other architectural practices that might help with communication.
   - ***3.9. Explain your Metaphors*** – When a new teammate is brought on board they will lack the shared experiences that went into creating the metaphor. Bring that person up to speed with diagrams and prose!
4. ***Case Study*** – To demonstrate the use of metaphors and the application of our guidelines, a case study with examples is presented from the authors' experience. (Pending space and flow, these examples may be interspersed with the guidelines in the written paper and will be explored in greater detail in our presentation.)
   - ***4.1. IVET Project Overview*** – Overview of the IVET project and Buzzhoney team including the makeup of the team, project background, and product under development.
   - ***4.2. Example Metaphors***
     - ***4.2.1. The "Bento Box" Metaphor –*** Example of a metaphor dealing with static structures and code organization.
     - ***4.2.2. The "Werewolf" Metaphor –*** This example shows how a good metaphor can be used to enhance team communication
     - ***4.2.3. The "Pilot-Navigator" Metaphor –*** Example of a metaphor dealing with dynamic structures.
     - ***4.2.4. The "Client-Server" Metaphor –*** The purpose of this example is to link current architecture practices with metaphors by showing the similarities between architectural styles and architecture metaphors.
   - ***4.3. Bad Examples –*** This section shares some of the bad metaphors created by the team and demonstrates common mistakes we made. Metaphors included: "Coat Hangers," "Model-View-Controller," and "Ordering Fajitas at a Restaurant."
5. ***Conclusions –*** Final thoughts and call to action for the continued use of metaphors and increased architectural thinking in agile development.

Authors' Background and Experience

Michael Keeling, Senior Software Engineer, Buzzhoney Email: mkeeling@neverletdown.net

Michael Keeling is a professional software engineer with over seven years of development experience. Prior to starting work at Buzzhoney, he was a systems analyst for Black Knight Technology for five years. At Black Knight he worked on a variety of software projects including analysis tools, real-time systems, and web-based applications. Keeling has a BS in computer science from the College of William and Mary and a Master's in Software Engineering from Carnegie Mellon University. His interests include the pragmatic application of software engineering methods, software architecture and design, and the human aspects of software engineering.

Other Publications:

- Jaspan, C., Keeling, M., Maccherone, L., Zenarosa, G. L., Shaw, M., *Software Mythbusters Explore Formal Methods*, IEEE Software, November/December 2009
- Keeling, M., *Process Affordances: Nudging Toward Change in your Organization*, Carnegie Mellon University MSE 20th Anniversary Conference, March 2010
- Keeling, M., *Put it to the Test: Using Lightweight Experiments to Drive Process Improvement*, 11th International Conference on Agile Software Development, June 2010

Mike Velichansky, Senior Software Engineer, Buzzhoney Email: michailv@gmail.com

Mike Velichansky started working in the IT industry as a copywriter, and only gradually transitioned into software engineering as various niches needed to be filled. Most of his time in IT has been spent developing small-to-medium scale marketing-driven websites for individuals and companies, but in more recent years his focus has become larger-scale web application development. Outside of IT, he's a published writer of fiction. As a writer, he was most interested in exploring the various ways methods of communication, written and oral, affect the development of software. He has a B.A. in English from the University of Maryland.