# Metaphors be with you!
(Metaphor System)

Dr. David West, Dion Stewart, Mathew Solano
Software Development Apprenticeship Program
New Mexico Highlands University

## Abstract

**One of the more unique aspects of XP compared to other agile methods was the explicit promotion of metaphor to an "official" practice. Unique: and immediately controversial. Few practitioners claimed to understand the point of metaphor and, indeed, it is no longer a separate practice in XP. Some agile proponents have noted general uses of metaphor in software development but there is still a need for more systematic discussion of how metaphor informs development. This paper will present one possible framework for such a discussion.**

## Background

Throughout this paper we will be using the term metaphor in a very loose fashion – as an umbrella for everything from a true metaphor to simile to analogy to story.

> *"Along the philosophical fringes of science we may find reasons to question basic conceptual structures and to grope for ways to refashion them. Old idioms are bound to fail us here, and only metaphor can begin to limn the new order."* [Quine79]

Each new software development project occurs at a fringe of understanding because the domain, the process, the problem to be addressed is, for the developers, initially, unknown. Metaphor is a useful, if not essential, tool for enhancing understanding and gaining knowledge.

Care in the choice of metaphor is important. Once coined, metaphors have a semi-independent lifecycle of their own.

> *"Metaphors can begin life as diaphors or epiphors and then change their status through usage and testing. Diaphors can become epiphors as their hypothetical suggestions find confirmation in experience or experiment. Epiphors become ordinary language when they are used so often that they express what the speakers now consider to become commonplace. When this occurs a new lexical [term] enters the dictionary."* [MacCormac85]

If the "hypothetical suggestions" of the metaphor are not confirmed the metaphor should be discarded. (Or take up residence in the world of poetry.) Occasionally an unconfirmed metaphor will continue to be used: either because it still has some pedagogical use for novices (the Bohr "solar system" metaphor of atomic structure) or because it is a kind of shorthand expression of a popular perspective – a paradigm in the

Kuhnian sense. When the latter circumstance prevails we can describe the phenomenon with the term "paraphor" – synthesis of paradigm and metaphor.

Metaphor is not just an explanatory device. There is strong evidence [MacCormac85, Cowan79, Lakoff99?] that the use of metaphor actually shapes the content and essence of the theory that employs them. In some very real ways – the metaphor you use (or fail to use) while developing a new system play a significant role in the success or failure of that system.

In 2001, metaphor was an official practice in XP and, because of how it was discussed came to be seen as singular in nature – the "system metaphor."

> *"Metaphor – Guide all development with a simple shares story of how the whole system works.*
>
> *Each XP project is guided by a single overarching metaphor. ... The words used to identify technical entities should be consistently taken from the chosen metaphor. ... The metaphor in XP replaces much of what other people call "architecture." [Beck01]*

In 2005 almost all agile proponents would agree with Beck that:

> *" ... metaphors are important. The language you use shapes the way you think which shapes how you program. Sharing metaphors does help a team communicate with each other and their users. However Metaphor is not an XP practice, any more than typing is a practice. Less, actually, since typing is a consciously learned skill. You learn metaphors from the moment that someone points to the picture in the board book and tries to convince you it's and airplane, maybe before. You know it isn't an airplane. You know they know it isn't and airplane. You learn that you use one thing to represent another. I don't think anyone can think abstractly without using metaphor. It can be very valuable to be aware of the metaphors you are using, sometimes even choosing them consciously. It is redundant to include metaphor as a practice, since you can't help having metaphors. That is why I removed it from the list of practices. It is still worth thinking about and talking about."* [Beck05]

We agree that metaphor is still worth thinking about and talking about, but we disagree that there cannot be a practice of metaphor. The appropriate use of metaphor, in our view, can be a consciously learned skill. We also believe that there is significant value in being aware of, and choosing, the metaphors you use.

**Default metaphors**

A significant majority of software developers, today, design and program under the influence of a set of "default" metaphors. For most this is not a conscious choice, merely

a reflection of the way they have been educated and trained. Prominent among these default metaphors:

Machine – a classic example of a metaphor that evolved to become a lexical term. Programming languages and operating systems are frequently discussed as "virtual machines" in that they represent the hardware at a higher level of abstraction. In languages like Smalltalk and Java the engine that allows execution is also called a "virtual machine" albeit with nuanced definition. Decomposition and design guided by this metaphor results in identification of lots of hierarchically nested synchronous and concurrent machines along with the control mechanisms to keep them interacting. Programming is metaphorically equated to "telling the machine what to do next."

Organization Chart – centralized hierarchical control, especially as embodied in the classic program structure chart with a topmost "master control module;" afferent, efferent, and transform modules below; and data and control signals passing among them. This metaphor is still embedded in most programming language structures in the form of a main() containing control flow logic calling subordinate functions.

Entity – a thing with characteristics the value of which must be remembered by the computer (or database

Objects are people too!