# Metaphor, Architecture, and XP

**David West**

New Mexico Highlands University

Las Vegas, NM, 87701

+1 505 454-3173

dwest@cs.nmhu.edu

**Abstract**

The concept of "system metaphor" as a kind of overarching story that guides development of XP projects at the system level is methodological weak point. This paper explores the idea of system metaphor as developed by various XP authors, and proposes a notion of architecture that seems suitable to XP.

**Keywords**

System metaphor, architecture, scaling

## 1 INTRODUCTION

At the recent XP Universe, one of the central topics of discussion - both formally and informally - centered on the ability of XP to scale - to be applied to large projects involving multiple developers. Issues of communication played a big role in this kind of discussion. Proponents of more "formal" methods, like RUP, frequently raised the "architecture" issue as a fundamental XP flaw, one that prevented solid communication and hence scaling.

There is certainly some merit in considering the needs of communication: within XP teams (users and managers considered part of the team for this discussion); among multiple teams; across the bounds of time; and, across the bounds of space. There is certainly a compelling desire for those communications to include some sort of gestalt perspective, which can provide both a focal point and a baseline understanding of the system. An "architectural model" is supposed to provide such a gestalt perspective and so, it would seem, be a desirable thing for XP to have.

And indeed XP does have "an architecture" - the *System Metaphor*. But is this sufficient? Critics obviously think not. If metaphor is insufficient, can we add something more "architectural" without violating the spirit and essence of XP.

## 2 METAPHOR

*"**System Metaphor** - A story that everyone - customers, programmers, and managers - can tell about how the system works." [Beck00: pp. 179]*

The total discussion of metaphor and architecture in, *eXtreme Programming eXplained*, occupies less than two of 190 pages. And yet, *"Architecture is just as important in XP projects as it is in any software project."* [Beck00: pp 113]

Martin Fowler notes the problematic nature of the System Metaphor:

*"Okay I might as well say it publicly: I still haven't got the hang of this metaphor thing. ...*

*Often people criticize XP on the basis that you do need at least some outline design of a system. XPers often respond with the answer 'that's the metaphor.' But I still don't think I've seen metaphor explained in a convincing manner. This is a real gap in XP, and one that the XPers need to sort out." [Succi01: pp 15]*

Mark Collins-Cope and Hubert Mathews propose that XP adopt a 'layered reference architecture [Succi01: pp 51-69], but this idea does not directly address the System Metaphor issue.

Newkirk and Martin finesse the issue entirely. [Newkirk01: pp18-19] They used a very familiar visual representation of the major components of a Web servlet system as their "system architecture.

William Wake provides the most extensive treatment

of metaphor in the XP series. [Wake02: pp 75-96] First, he suggests that the traditional role of architecture is less relevant in XP.

*"Extreme Programming (XP) places less emphasis on up-front architecture than other methods because architecture has less impact in XP: XP programmers work to keep the system flexible. XP says "embrace change," whereas architecture-driven approaches say, "Some things are hard to change, so plan the skeleton first."*

He then goes on to suggest that a specific architecture, of sorts, is created anyway - especially in the "spike" and "first iteration" elements of the XP process. [Wake02:pp 76] And, quoting Kent Beck, "*The first iteration must be a functioning skeleton of the system as a whole*." [Wake02: pp78] Wake then suggests that a proper system metaphor supports four basic aspects of system building: common vision, shared vocabulary, generativity, and architecture. Architecture, *"...shapes the system by identifying key objects and suggesting aspects of their interfaces."* [Wake02: pp85-86]

Despite these efforts we are left in the same position as Martin Fowler - still absent, *"the hang of this metaphor thing."*

## 3 PRESUPPOSITIONS

The need for an architecture - or the XP alternative, System Metaphor - seems to be a given.

If, "architecture is just as important in XP projects as it is in any software project," it must be presumed that the importance is for reason(s) other than those espoused by traditional software development methods. It is useful to ground those reasons, where possible, in the XP Four Values.

**Communication**. Keeping everyone informed about what is going on is essential to XP. Communications take many forms and almost every XP practice incorporates and reinforces some form of communication. A system metaphor or architecture should function as a means of enhancing communication. A way of keeping track of all the stories - and relationships among stories - that actually drive development activities.

Some ad hoc ways of providing this coordination are evident in XP. Bulletin board with post it notes or story cards are but one example. A way of enhancing these ad hoc approaches would be very useful.

**Simplicity**. Architecture is a tool, nothing more. As such, you should be able to create one simply and easily without special skills or extensive knowledge of modeling syntax. People should be able to modify it easily and simply and, conceptually, it should be comprehensible at a glance and understandable with minimal perusal.

**Feedback**. Architecture should be no more static than the system which it models. In addition to allowing simple modification, the architecture should capture state or time in some fashion - even to as minimal a level as showing the status of individual actions, stories, or subsystems.

**Courage**. Here I am taking some liberties and pushing the envelope of what courage is all about. The form of the architecture should make a bold and unmistakable statement to the effect, "This is an XP project!" It should be unique and interesting so as to draw the attention and the questions of those that should know about XP but, as yet, do not.

**Evocative**. (Not one of the four XP values.) Traditional software models, including architectures, strive to be representational in essence (e.g., "this box stands for this block of code"). But, representational models suffer from their inability to provide a one-to-one mapping. So the representations are really abstractions. The "map is not the territory" (Korzibski). And most of the interesting and critical details escape the architecture.

Human beings are not limited to representational knowledge - even though one of our main modes of communication - the written language - is purely representational. We also embody experiential and kinesthetic knowledge. And we thrive on associational or evocative knowledge.

An interesting thing about evocative knowledge - it can be very complex. Each frame of a Ridley Scott movie exhibits immense complexity in terms of the things visible in that frame. They are powerfully evocative, interesting, and compelling. In contrast, a typical cel from a modern Saturday morning cartoon is almost devoid of information. They are quickly boring to all but the most childish viewers. This means that our architectural model can be complex in appearance without violating the simplicity requirement noted above.

## 4 XP ARCHITECTURE

Figure One (at end of paper) is a digital photo of a Thangka painting on the wall of my office. Depicted on this painting is an evocative model of Tibetan Buddhist cosmology and philosophy. Even a cursory glance reveals that it has some kind of structural organization, significant amount of detail, and lots of interesting and unusual images.

What you see on the painting is only marginally representational. Each icon and each organizational segment of the whole can be seen as representing a particular deity or circumstance but that is almost coincidental. The real purpose of the painting, and each individual element, is to evoke memories - primarily memories of stories you were told about Tibetan Buddhism.

I propose that this kind of painting be the foundation for an XP architectural model.

Essential elements of the painting that would have equivalents in the XP architectural model include:

- A center circle with icons evoking the driving forces behind the system under development. In the Thangka these are attachment, greed, and anger. In a system model these might remind us of the importance of money (almost inevitable), a particular client that will pass final judgment on our work, a customer, or a service.
- A large segmented circle, each segment representing some kind of partitioning of the system - preferably isomorphic with the natural segmentation of the domain. In the painting these are the various realms of existence, e.g. heaven, hell, material world. In our model these might represent realms like order entry, inventory, accounting, manufacturing; or, segments of a smaller scale system - data entry, backend processing, backup and recovery, etc.
- Icons of various sorts arranged in a tableau evocative of the stories that relate those icons to each other. Each icon evokes a specific story or set of stories about a particular element of the overall system. Icons can appear in more than one segment.
- A narrower outer circle, also segmented, with each segment representing a stage in the "circle of life." Each segment of this circle could represent a processing stage or step in a business cycle..
- Finally, outside the circle - icons that recall stories about outside influences. Things or forces or people that can affect our system but are outside the scope of what we can actually build.

Dynamism can be added in many different ways. If each icon evokes a particular story and the work required to realize that story, then the orientation of the icon (down = not started, right angle left = in progress, vertical = done, right angle right = abandoned dead end) can be used to depict status. A quick glance at the model reveals a rough approximation of total effort and total achievements.

The Mandala Architecture works. The model has been used as a way to train potential architects in the creation of richly detailed gestalt models of complex processes. The actual exercise began with about 30 minutes discussion of objectives, issues of complexity and modeling, purposes of gestalt models and examples of gestalt models. A ten-minute explanation of the Mandala Architecture preceded the exercise. Three person teams were given blank overhead transparencies and four colored markers and 30-40 minutes to talk about and construct their Mandalas.

Every team was able to construct and explain a Mandala. Sophistication varied, of course, with some icons being crude caricatures and others exhibiting some degree of artistic talent. But the quality of the icons did not matter. Explanations of the mandala contents required the telling of stories. Since all of the participants were from the same company they had a shared "company lore" that they could refer to with shorthand descriptions.

In subsequent years I used this same exercise in more than fifteen courses. Most were directed towards professional software developers or students in a graduate software engineering program. At least twice, however, business managers and executives constituted the audience. In all cases the results were amazingly successful. Actual Mandala models ranged from hand drawn transparencies with 15-20 icons to a really complex model created by a student using Visio with over a hundred icons. There was no marked difference in the overall sophistication of the models created by managers and by professionals. In the few cases where both a manager and a software developer created models of the same domain - the degree of convergence in content was extremely high (about 90%) and the stories evoked when explaining the models were almost identical.

On two occasions when this material was part of a semester long course I had students present their models about a third of the way through the semester. Roughly four weeks later I asked students to present the mandala models created by *other* students. These re-presentations were almost as detailed as the originals. The recall exhibited in this exercise was roughly triple the level of performance that the same students exhibited in pop quizzes about other models (classical DFD, ERD, etc.)

## 5 CONCLUSION

The System Metaphor concept in XP is a potential weak point and is not very well understood.

Mandala Architectures provide a visual metaphor of potentially rich complexity without contravening any of the XP Four Values.

The simplicity of creation, depth of explanation, degree of recall, and richness of story telling have all been demonstrated when this architectural approach was used with both professional developers and business managers.

## 6 REFERENCES


[Beck00] Beck, Kent. Extreme Programming explained. Reading, Mass: Addison-Wesley Longman, Inc. 2000.

[Beck01] Beck, Kent and Martin Fowler. Planning Extreme Programming. Reading, Mass: Addison-Wesley Longman, Inc. 2001.

[Newkirk01] Newkirk, James and Robert C. Martin. Extreme Programming in Practice. Reading, Mass: Addison-Wesley Longman, Inc. 2001.

[Succi01] Succi, Giancarlo and Michele Marchesi. Extreme Programming Examined. Reading, Mass: Addison-Wesley Longman, Inc. 2001.

[Wake02] Wake, William C. Extreme Programming Explained. Reading, Mass: Addison-Wesley Longman, Inc. 2002.


Figure One - "Wheel of Life " Thangka