The eXtreme Programming (XP) Metaphor and Software Architecture

James Herbsleb, David Root, and James Tomayko

August 2003
CMU-CS-03-167

Also published as CMU-ISRI-03-103

School of Computer Science
Carnegie Mellon
Pittsburgh, PA, USA

Abstract

The Metaphor is intended to contribute to the Agile Programming value of communication. Previously, some of the author studied the Metaphor as a means of communication among team members and between them and clients. This paper examines the Metaphor's contribution to the software architecture. Both experiments seem to reveal that the Metaphor has poor effectiveness.

Keywords: software architecture, extreme programming

Earlier this year, two of the authors did a study of the effectiveness of the XP Metaphor, concentrating on evaluating it as a communications mechanism and later we planned examining it in developing the software architecture of a project [1]. The "metaphor" is the practice of agile processes most ignored by practitioners. Almost all agile methods explicitly cite "communication" as a key value. This is meant to be communication about the software among the development team, as well as among and with the clients.

The use of a metaphor is a powerful tool to achieve this [2]. There is a "naïve metaphor," which is a metaphor very close to the actual product. The financial planner product described in this experiment is an example. Also, there is a more complex, natural language, metaphor.

The metaphor has two purposes. The first is the communication described above. A user ought to have an easier time speaking and giving examples about an "accountant," than about Quicken. A second reason is that the metaphor is supposed to contribute to the team's development of software architecture. Ken Auer and Roy Miller describe the XP metaphor as being "analogous to what most methodologies call architecture"[6]. The purpose of this paper is to study the effectiveness of the metaphor in this light..

Kent Beck, originator of eXtreme Programming (XP), recently felt the need to justify the practice in a recent keynote talk, and offered, depending on the outcome of an audience vote, to "just shut up about it" [3]. While such a vote makes for good theater, practitioners would be better advised to base such a decision on evidence of effectiveness, such as these two papers.

A metaphor is meant to be agreed upon by all members of a project as a means of guiding the structure of the architecture [4]. As an example, "check writer" is a poor metaphor for financial software tools like Quicken, but "accountant" reveals more functionality to users and provides more guidance for high-level design.

There are two times where the metaphor can be tested for effectiveness: soon after it is developed, to see if it helps with design and with communication, and when the architecture is finally developed, to see how the metaphor influenced it. This study focuses on the latter. Two authors examined the former in [1].

The Use of the Metaphor to Explain the Architecture

This is how information relative to the architecture was gathered:

Each team was asked to develop a metaphor for their software (see attached assignment, Appendix A). They were asked, as part of the assignment, to keep track of all of the metaphors they considered, and the amount of time they spent on the assignment (neither of which would affect their grades). This data was more extensively used in [1].

Toward the end of the course, students were given individual semi-structured interviews to find out if their descriptions of the metaphor matched that of the other members of their team, and the extent to which they were using their metaphor as a means of communicating their project. They were also asked to sketch the architecture. An analysis of how these sketches matched their final team architecture is the subject of this paper. The key questions are:

Cost: What was the cost (in time) for generating a metaphor? (Answered more fully in [1].

Utility: Were the metaphors considered to be useful in the design process? Were they considered to be useful for developing architecture?

Experimental Context

The initial and later experiment was conducted in the context of a software development Studio course and a software requirements course. There were 27 software engineers, seven architects, and one civil engineer in the requirements course. For the computational architecture majors, this is a required course; for the engineers, it is an elective. Some of the software engineers are taking a program that requires them to work on a project during their entire year of study in school (the Studio project course). All Studio projects have actual clients with real needs, specific deliverables, and regular meetings with the client. Typically these projects are assigned five to six Masters of Software Engineering students, not all of whom were in the software requirements course. Some of these members were also queried about use of a metaphor on the Studio project.

Experimental Design

The student teams had an assignment in their requirements engineering class to develop a metaphor for their project, and to track how long it took them to develop one. The teams had two weeks to complete the assignment. (The exact assignment is reproduced in Appendix A.) Each team submitted a written statement of their final metaphor, along with a description of any false starts and the total time they spent on the assignment. The interviews were recorded on paper, and the early architecture collected on the back of the sheet.

Even though there was no formal instruction in metaphors or how to create them, the teams had little difficulty developing metaphors that seemed, to them and to us, meaningful and appropriate. To our knowledge, there are no suggestions in the agile literature that formal instruction in metaphor is needed, so this procedure seems close to what one would expect in industrial settings.

Several weeks after the assignment was turned in, the authors met with each team member individually to conduct semi-structured interviews according to Appendix B. The interviews began with fairly general, open-ended questions about what the team was building, how it would work, and what the main components would be. Next the team member was asked to describe the metaphor developed by the team, with follow-up questions as needed until the interviewer understood the metaphor and how it applied to the project. Finally we asked each interviewee to respond to six statements about the utility of the team's metaphor with respect to coming up with a design, communication among team members, communication with the customer, and whether they recommend metaphor development for future classes. The interviewee was given five possible responses: strongly agree, agree, neutral, disagree, or strongly disagree. The interviewee was shown both the statements and the possible responses.

Answers to a slightly modified set of questions and an architectural sketch were extracted from a single Studio member in each team who did not take the requirements course. The questionnaire used for this is in Appendix C. If the metaphor has any use, then we think it will show up in a comparison of the architecture drawings made by team members who did the assignment, and those who worked with them.

Results

The Metaphors

The metaphors generated by the five Studio teams are presented in the table below.

Project	Metaphor	Explanation
Wrist camera	Portrait studio	The software has the capability for transferring images from one device (e.g., PDA, PC) to another, and some image processing capabilities. It is much like a portrait studio, where a camera takes a picture, which is developed, retouched, printed, and distributed.
Wrist camera	Cities and Towns	(same assignment as above). Larger, more capable devices are like cities, in which many services are available. Smaller, less capable devices are like small cities, or even villages, where fewer services are available. Transfer of files is like a train moving from one municipality to another.
Financial planner	Human financial planner	The software follows a specific 5-step method for preparing a financial plan. The metaphor is that the program will behave, as would a human planner.
Department of Transportation (DOT) web site	TurboTax	Allows a person to register cars, transfer titles, and perform other standard DOT functions. Will be driven by a script that asks questions meaningful to users, automatically populate new forms with data, in a way similar to TurboTax.
Ford Motor Company software architecture tool	C compiler	A tool is being developed to combine architectures of components into one major architectural artifact.

A way to test the usefulness of the metaphors is to see the extent to which the metaphor is reflected in the architecture. We had each student indicate through drawings and words on the back of the interview sheet the nature and form of the architecture as they imagine it. The final architecture is not done, but nearly all of the students were able to draw a "preliminary" architecture when requested.

These will be compared to the final architectures. Our preliminary results showed that 4 persons could not draw architecture.□For the others, we looked for evidence of at least one word or concept either in the drawing itself (including text labels), or in the explanation they gave of the drawing as they described it to us.□While 6 showed some evidence of the metaphor, 14 showed no evidence of the metaphor at all.
On the other hand, the original architectural drawings for the projects are nearly identical. The drawing made by the other team members later had all the components mentioned by the students in the first round, plus some additional ones. This might represent additional understanding, or that the student in question already had, or was taking concurrently, a software architecture course.

The results from examining the architecture drawings of the other team members are similarly unsupportive of the metaphor. Only in very few cases did concepts and terminology from the metaphors seep into the anticipated architecture or the students' explanations of their architectures.

Conclusion

These studies indicate that natural language metaphors are relatively useless for either fostering communication among technical and non-technical project members or in developing architecture.

Admittedly this is a relatively small sample size mostly comprised of relatively experienced professional software engineers (minimum of 5 years industry experience) that are highly screened students in the Carnegie Mellon program. However, the anecdotal data from the field almost universally supports these conclusions.

Why is the metaphor so difficult? Mostly because originators are engineers, and they do not have the proper skill set. Add to this rather poor presentation of the metaphor [4, 5 , 6,, 7], and the inability for non-technical people to add to it, and the results are not surprising. Many teams fall back to the "we built one like this last July" type of metaphor,

References

[1] Herbsleb, Jim and Jim Tomayko, "How Useful Is the Metaphor Component of Agile Methods? A Preliminary Study," CMU-CS-03-152, 2003.

[2] Bipin Indurkhya, *Metaphor and Cognition*, Kluwer, 1992.

[3] http://oopsla.acm.org/fp/files/spe-metahpor.html

[4] Beck, Kent*,* *Extreme Programming Explained*, Addison-Wesley, 2000.

[5] Newkirk, James, and Robert C. Martin, *Extreme Programming in Practice*, Addison-Wesley, 2001.

[6] Auer, Ken, and Roy Miller, *Extreme Programming Applied,* Addison-Wesley, 2002.

[7] Wake, William C., *Extreme Programming Explored,* Addison-Wesley, 2002.

Appendix A: The Metaphor Assignment

Due: Friday, 1 November 2002

Making a Metaphor

In agile methods, especially eXtreme Programming, a *metaphor* of the project is developed to help guide a team toward a good architecture and a clearer way to discuss the structure of the software with the client.

For more information, read Kent Beck's *Extreme Programming Explained* [Addison-Wesley, 1999, Ch. 10] or look at web links. Basically, an 'automated checkbook' is a poor metaphor for Quicken, as it does much more. An 'automated accountant' is better, as it captures more functionality.

Try to develop a metaphor for your project among your team. Submit the final metaphor and all the drafts, false starts, etc. Also, keep careful track of the time it takes to build the metaphor, and submit that figure also.

Appendix B: The Interview Questionnaire

[The first several questions were designed to see if the students spontaneously used the metaphor to explain their project. In all interviews, there was at least one person, either a note-taker or the interviewer, who was unfamiliar with the projects. The interviewee was asked to give a quick description of what they were building, ostensibly just to provide some background.]

Background
First, could you give us a little background about your project? Can you describe very briefly what is it that your group is building?
What are the main parts?
How it will work?

Metaphor
Would you describe the metaphor your group came up with?

Utility of metaphor
For the next several questions, I'll read you a short statement and ask you to tell me how much you agree or disagree with the statement, from strongly disagree, disagree, neutral, agree, or strongly agree. [Show them a piece of paper with the rating scale on it.]

The metaphor has been helpful in figuring out the overall design of the program.

The metaphor has helped the team find a common vocabulary.

We often use the metaphor in conversations with each other.

We often use the metaphor in conversations with our customer.

The metaphor is useful in helping everyone reach agreement about our requirements.

I recommend that future classes create metaphors for their projects.

Appendix C

1. Have the members of your Studio project team used a metaphor other than the software project objective itself to discuss the software? If so, what was it?

2. On the reverse side of this sheet, draw and label the software architecture, as you presently understand it.

Utility of metaphor for developing the architecture

For the next several questions, read the short statement and indicate how much you agree or disagree with the statement, from strongly disagree, disagree, neutral, agree, or strongly agree.

The metaphor has been helpful in figuring out the overall design of the program.

The metaphor has helped the team find a common vocabulary.

We often use the metaphor in conversations with each other.

We often use the metaphor in conversations with our customer.

The metaphor is useful in helping everyone reach agreement about our requirements.

I recommend that future classes create metaphors for their projects.