Rearchitecting a Software Platform

A Beginner's Case Study with Dialog Mapping

Eugene Eric Kim <eekim@eekim.com>

June 11, 2002

Abstract 1 (01)

In February 2002, I had the opportunity to use Dialog Mapping extensively on a large consulting project for TeamSphere Interactive. Despite my lack of experience, I found the technique extremely valuable, and I believe that it significantly influenced the direction and outcome of the project. This paper relates my Dialog Mapping experience at TeamSphere and the lessons learned along the way. 1A (02)

The Project 2 (03)

TeamSphere Interactive is a small company in San Francisco, California that specializes in online gaming communities. TeamSphere hosts online tournaments and provides other community-related services to client companies in the gaming industry using a software platform called Eventricity, which was first designed and implemented in late 1999. Since the company's inception, TeamSphere's engineers had been customizing and extending Eventricity to support client needs, and they were beginning to push against the limitations of the original architecture. 2A (04)

In February 2002, TeamSphere hired me as a consultant to rearchitect Eventricity. I had worked with TeamSphere in early 2000, managing the development of an early version of its product, so I was already familiar with both the software's architecture and the company's employees. 2B (05)

In order to understand the concerns of the employees throughout the company (production, engineering, QA, and sales) and the state of the existing system, I organized an all-hands meeting and Dialog Mapped the session. From the discussion that ensued, I realized that my task would extend well beyond proposing a new software design. Every employee agreed that the system needed to be rearchitected, but each individual had also come to their own conclusions -- colored by their own needs and experiences -- as to how the system needed to be improved. Because there had been limited time to interact about anything but the short-term tasks at-hand, as is typical at small, startup companies, the employees had not had the opportunity shared and discussed their views with each other. 2C (06)

The majority of the new technical architecture and overall development process already existed in the minds of those already at the company. My challenge, then, was not to derive my own solution independently, but to capture and synthesize the solution that existed collectively in all of our heads. 2D (07)

By reframing my task this way, it became clear that Dialog Mapping could play a much greater role than simply capturing initial requirements. Over the first month of my three month contract, I facilitated seven Dialog Mapped meetings. These sessions consisted of between two to six people, including myself. I refactored the map often, most significantly after the first meeting. 2E (08)

First Meeting 3 (09)

My goal for the first meeting was to capture the issues that individuals felt were most significant. Based on informal discussions with the employees prior to the meeting, I knew that the list of issues would be extensive. In order to get some idea of what the Dialog Map might look like, I gave each employee homework a week in advance, which I called the Jeopardy Assignment. I asked everyone to brainstorm a list of issues they felt were important, phrased as questions. I had everyone e-mail those questions to the rest of the group prior to the meeting, in order to reduce repetition and possibly jog the memories of others. 3A (010)

The homework was a success. We received a total of 99 questions, all of which I considered important and useful to capture. The evening before the first meeting, I mapped the first 10 questions, mainly as a way to anticipate the overall structure of the map. 3B (011)

At the meeting, I reviewed the results of my work with the group so that they could understand how the Dialog Mapping process worked. I then began our session in earnest, encouraging people to discuss the issues they felt were important, using the list of questions as a guide if they wished. 3C (012)

The actual facilitation process went well. The discussion was lively, but constructive, and the Dialog Map played an active role in guiding the discussion. I believe that a major reason the process worked well was that the room was small and intimate, and the shared display was large and constantly in everyone's field of vision. Having the questions ahead of time also helped the process go smoothly. Because I already had an idea of what people would discuss, it required less mental effort to determine which nodes to create, what to name them, and how to link them. As a result, I could keep pace with the discussion without having to slow it down much. *3D* (013)

The resulting Dialog Map, unfortunately, was not very good. To understand why requires some understanding of my own background. I first learned about Dialog Mapping from one of Jeff Conklin's two day workshops, which I attended in July 2001. All of the attendees had the opportunity to try the technique and get feedback from Conklin, and by the end of the workshop, I was comfortable enough with the technique and the tool (QuestMap) to try it in a real group setting. *3E* (014)

I put this new knowledge to use immediately. That same month, I Dialog Mapped two small meetings, each of which lasted two hours and which consisted of six or seven people. These meetings presented a relatively low degree of difficulty to me, because they were expository discussions consisting mostly of questions and ideas directed at one individual, and because I was knowledgable about the content of the discussions. Also, although several of the topics discussed were definitely wicked problems, the discussions centered around a framework that had already been developed to address these problems. Even with all of these factors in my favor, I still found the process nerve-wracking at times. Nevertheless, the meetings went well enough and the resulting maps were good enough to make me confident that I could put this technique to good use, even at a novice level. *3F* (015)

Unfortunately, I was still unquestionably a novice, as the results of my first session with TeamSphere showed. The first hour of the 90-minute session went smoothly, but I began having trouble soon after. The problem was that I was using only one map, and it was becoming crowded and disorganized. I was having difficulty determining where I should add new nodes, and after I decided, I spent too much time moving old nodes around so that I could fit the new ones. *3G* (016)

After the meeting, I reorganized the map and added all of the questions from the list that were not discussed at the meeting. After refactoring, the map consisted of 187 nodes distributed among six trees, with each tree containing a mean of 26.71 nodes. 96 of all nodes were issues, 79 were ideas, and the rest were arguments. *3H* (017)

A Compendium Epiphany *4* (018)

After this first meeting, I reported my results to Conklin's Dialog Mapping discussion list, and complained about the inadequacies of the tool, QuestMap. I soon realized that the problems with my map had more to do with my inexperience than with the inadequacies of the tool. *4A* (019)

I came to this conclusion, because shortly thereafter, I had the good fortune of watching an experienced facilitator at work. A nonprofit organization with which I am involved held a small, internal workshop regarding its mission and strategy, and we hired Conklin to facilitate. This was the first opportunity I had had to watch Conklin work in a noncontrived setting. *4B* (020)

I learned several tricks by observing Conklin, the most significant of which was that combining template maps with a system of codes and transclusions greatly increased the effectiveness of the map. I later learned that these principles are the cornerstones of the Compendium methodology, of which I had no prior awareness. [Conklin et al 2001] [Selvin et al 2001] *4C* (021)

Meetings will naturally flow around certain topics. Facilitators can take advantage of this fact by preparing a template map before the meeting that represents the topics to be discussed. One side-effect of using a template is that you use a new map for distinct topics. When there is overlap between maps, you can use transclusions to share nodes. Finally, you can insert a code in the details of a node so that you can easily locate nodes later. For instance, nodes that represent software requirements could include the code "$requirement". Later, you could create a new list view called "Requirements", and populate it by searching for all nodes with the proper code. *4D* (022)

The TeamSphere problem could naturally be divided into several topics. We needed to document the current architecture, we needed to discuss the new architecture, we needed to create a migration process from the old to the new system, and so forth. I created new maps for each of these documents, organized these maps into a high-level hierarchy of trees, and reorganized the original map accordingly. *4E* (023)

One reason I had not used multiple maps the first time around was that I was paranoid about losing linkages. Transclusions

appeased this concern. If I didn't want to lose the context of a tree, I could always transclude parts of that tree into the new map. As it turned out, the final Dialog Map used only two transclusions out of a total of 526 nodes distributed among 24 maps. I may have underutilized transclusions, but I believe that ultimately, the problem was not wicked enough to merit wide use of them, as I explain in more detail later in this paper. The more important lesson was that my fretting about losing linkages was misguided, and that using multiple maps in the first place would have helped me tremendously. Subdividing the maps this way markedly improved the organization and navigability of each map. From that point forward, I rarely got lost in a map, and I rarely needed to spend extensive time moving old nodes around so I could fit in new ones. *4F* (024)

The final Dialog Map consisted of 24 maps and a total of 524 nodes. Each map had a mean of 21.92 nodes (18 median) distributed among 3.58 trees (2.5 median). Each map contained a mean of 6.58 issues (5.5 median) and 12.42 ideas (9.5 median), with the remaining nodes consisting mostly of arguments and some maps, lists, notes, and decisions. The largest map had 50 nodes (6 issues, 42 ideas, 2 arguments) and the two smallest had 10 (2 issues, 8 ideas in one map, 2 issues, 7 ideas, and 1 argument in the other). Each tree had a mean of 6.24 nodes (3 median), 1.85 of which were issues (1 median) and 3.48 of which were ideas (1 median). *4G* (025)

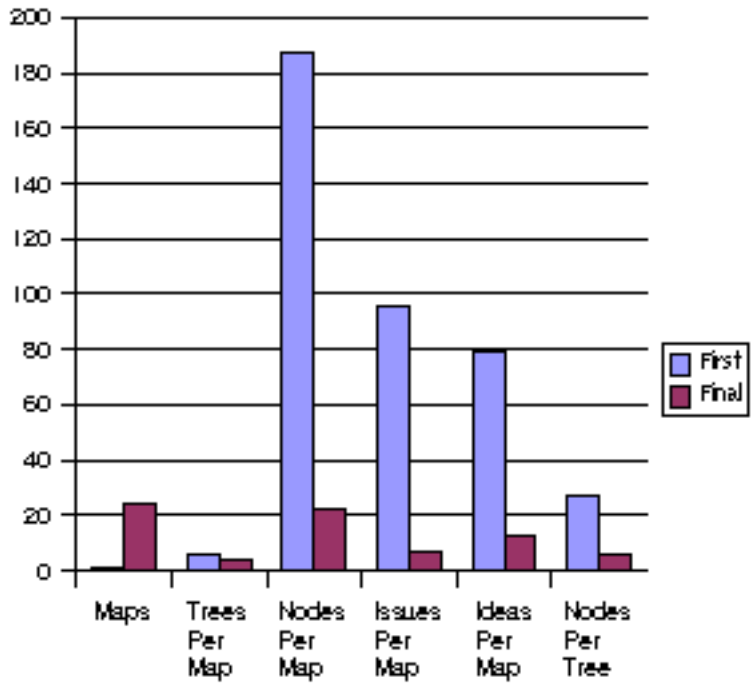

Figure 1. A comparison of the distribution of nodes between the first and final versions of the TeamSphere Dialog Map. *4H* (026)

Figure 1 compares the node distribution of the first and final maps. The size of the trees in the final map are significantly smaller than in the original, which naturally made for a more manageable map. The original map had about a 1:1 ratio of issues-to-ideas, whereas the final map had an issues-to-ideas ratio of almost 1:2. *4I* (027)

What We Gained *5* (028)

Conklin defines wicked problems as having several attributes, one of which is that the problem itself is not well-understood. [Conklin 2001] We were not dealing with wicked problems at TeamSphere. Our goal was to improve the software's architecture and the underlying organizational processes so that TeamSphere could better serve its clients' needs. These problems were well-understood, but they were also large and complex. *5A* (029)

The resulting Dialog Map served two important purposes. First, it effectively captured the breadth and depth of the problem. In order to come up with proper solutions, we had to document all of the pertinent issues comprehensively, so that we could be sure to address (or explicitly avoid) all of them. The graphical IBIS representation did an excellent job of capturing the essence of the conversation in easily readable form. Throughout the process, we used the map as a checklist to make sure that

the system and processes we were developing were meeting all of the necessary requirements. *5B* (030)

Second, the map served as a catalyst for effective, constructive discussion. Employees are naturally territorial about their ideas and interests, and the stronger personalities in a small group can often intimidate their peers into accepting their ideas. The Dialog Map depoliticized our discussions. All of the ideas were recorded anonymously on the map. Ensuing discussion centered around the ideas themselves, rather than the people. I observed several instances where one party considered ideas proposed by their peers more seriously than they would have had it been a one-on-one discussion without the map. *5C* (031)

Additionally, the IBIS grammar served its purpose of encouraging the participants to consider underlying questions. In a one-on-one session I had with TeamSphere's CEO, I asked him to brainstorm all of the marketing challenges regarding the product without immediately proposing solutions to them, and I would capture his thoughts using QuestMap. After about 10 minutes, he complained that my request was impossible, and that the map was making it difficult for him to focus exclusively on listing issues. Vital questions were literally staring out at him from the map, and he found it very difficult to avoid them. I told him that was okay, and to just go with the flow. *5D* (032)

This process was invaluable, because he kept returning to the same set of core issues, and even though he had not explicitly said so earlier, it was clear that these issues were of foremost significance in his mind. In particular, that process allowed us to have a vital insight that no one had previously made, which led to an important shift in the direction of the overall project. *5E* (033)

Thoughts on Dialog Mapping *6* (034)

My work at TeamSphere essentially began as a software project, and quickly morphed into a group facilitation exercise. Consequently, Dialog Mapping proved to be an enormously valuable tool. While we benefitted in many ways from using Dialog Mapping, I also gained a deeper understanding of the technique itself. *6A* (035)

I had understood the utility of a shared, live display even before I learned about Dialog Mapping, and I used it even when not Dialog Mapping. On several occasions, we abandoned QuestMap for the whiteboard, collaboratively developing tables and object models. For less discussion-oriented meetings, I used a live, public display to capture meeting minutes and action items in real-time. I encouraged TeamSphere engineers to practice Pair Programming, a collaborative software development technique where two programmers share a single keyboard and monitor. *6B* (036)

Clearly, IBIS is not the end-all and be-all of grammars, and there is a right time and place to use it. That being said, I cannot overemphasize the value of IBIS's simplicity and question-oriented approach when working towards a collaborative solution to large, complex problems, wicked and otherwise. At TeamSphere, I never needed to explain the IBIS grammar. They understood it immediately, and followed the map without distraction. Additionally, the grammar provides an ideal balance between transcription and interpretation. Several people at TeamSphere expressed enthusiasm for how well the Dialog Map represented their overall thinking in a relatively small space. Finally, thinking in terms of questions helps prevent narrow thinking, and improves the overall dynamics of a meeting. Placing the burden of identifying the questions on the facilitator allows the meeting participants to benefit from the technique without the cognitive overhead. *6C* (037)

In order to publish the Dialog Maps, I used a tool that I wrote called perlIBIS, which can convert QuestMap files into an HTML outline. I found that reading the Dialog Maps in outline form was generally easier than reading them in QuestMap. I believe the reason for this is that our maps translated easily to outline form, and that having to scroll vertically is easier than having to scroll horizontally and vertically. However, QuestMap's graphical depiction of the maps was superior to the outline form for discussion-oriented navigation, because the second dimension serves as an additional reference for locating topics. Ideally, a Dialog Mapping tool would have the native capability to display diagrams as both two-dimensional maps and one-dimensional outlines. *6D* (038)

Finally, while I no longer consider myself a novice at Dialog Mapping, my memories of frantically moving nodes out of the way so I could add new ones to my overcrowded map are all too fresh. Nevertheless, I found my first attempt to be immediately valuable, and my skills improved with each subsequent attempt. Additionally, refactoring is an important practice, and maps can be and should be improved. While the learning curve is not trivial, it is not steep either. Like any useful endeavour, you must practice Dialog Mapping in order to master it. Unlike many similar endeavours, your results will be useful even if you have never Dialog Mapped before. *6E* (039)

Acknowledgements *7* (040)

I am grateful to Justin Lin and his hardworking team at TeamSphere for being tolerant and enthusiastic about embracing new

ideas and tools in a project of great importance. Many, many thanks go to Jeff Conklin, who graciously allowed me to participate in his workshop and learn about this wonderful technique, and who has been a constant source of advice and encouragement. *7A* (041)

About the Author *8* (042)

Eugene Eric Kim is an independent consultant, writer, and programmer. He has been an active contributor to Doug Engelbart's Bootstrap Alliance and Open Hyperdocument System. As the Foundry Guide for Sourceforge's Distributed Computing Foundry, he actively facilitates collaboration between open source software communities involved with distributed computing projects. Eugene has consulted for a number of organizations, including PricewaterhouseCoopers, and has written for a variety of publications, including *Scientific American*. Previously, he served as the Senior Technical Editor for *Dr. Dobb's Journal*, the leading magazine for software developers. He is the author of *CGI Developer's Guide* (Sams.net 1996), and is currently writing a book on the history of free software. Eugene received an A.B. in History and Science from Harvard University. *8A* (043)

References *9* (044)

[**Conklin 2001**] Conklin, Jeff. "Wicked Problems and Fragmentation." 2001. http://cognexus.org/wpf/wickedproblems.pdf *9A* (045)

[**Conklin et al 2001**] Conklin, Jeff, Albert Selvin, Simon Buckingham Shum, Maarten Sierhuis. "Facilitated Hypertext for Collective Sensemaking: 15 Years on from gIBIS." KMI-TR 112. September 19, 2001. http://kmi.open.ac.uk/tr/papers/kmi-tr-112.pdf *9B* (046)

perlIBIS. http://www.eekim.com/software/perlIBIS/ *9C* (047)

QuestMap. http://www.softbicycle.com/questmp.html *9D* (048)

[**Selvin et al 2001**] Selvin, Albert, Simon Buckingham Shum, Maarten Sierhuis, Jeff Conklin, Beatrix Zimmerman, Charles Palus, Wilfred Drath, David Horth, John Domingue, Enrico Motta, and Gangmin Li. "Compendium: Making Meetings into Knowledge Events." *Knowledge Technologies 2001*. March 4-7, 2001, Austin, TX. *9E* (049)

TeamSphere. http://www.teamsphere.com/ *9F* (050)