The Making of *Trigger* and the Agile Engineering of Artist-Scientist Collaboration

Francis T. Marchese
Department of Computer Science, Pace University, New York, NY 10038
fmarchese@pace.edu

Abstract

The agile process approach used in software engineering was adapted as a foundation for the management of the multimedia art installation project titled "Trigger." By beginning with requirements engineering and risk management assessments, instead of rigid planning, the project was designed with flexibility that allowed for iterative adjustment. The residency period drew the artist, Jody Zellen, and the computer scientists together for agile learning by the processes of doing and testing. This style of management allowed the group to act decisively with incomplete information, thereby successfully finishing the project in time to meet a rigid deadline.

1. Introduction

On October 28, 2005 a site-specific, sensor-activated, immersively projected, interactive art installation called Trigger [1] opened in the Pace Digital Gallery in downtown New York City. Trigger was the result of collaboration between the Los Angeles artist Jody Zellen and computer scientists from Pace University's Center for Advanced Media (CAM) as part of its first artist-in-residence. It was also a first for CAM, because it brought together an artist and scientists from different parts of the country, to design and build a project utilizing untried technology, under time and budgetary constraints. In order to ensure that the project would be successfully completed on time, it was necessary to recast the project's structure and process, utilizing software engineering techniques that systematized the practice of designing and building software [2] - [3]. Agile software engineering methods [4] – [9] were selected because they deal directly with ambiguity, uncertainty, and change - issues that are intrinsic to scientific and artistic enterprises - by creating an environment that embraces evolution, communication, and adaptability.

In subsequent sections we present a description of *Trigger* from the software engineering perspective. The next section presents a brief introduction to software engineering.

2. Background

Software engineering is a systematic approach to analysis, design, implementation and maintenance of software, that is, the application of engineering to software [2]. Software engineering covers not only the technical aspects of building software systems, but also management issues, such as directing development teams, scheduling, and budgeting.

A software engineering project follows a lifecycle that segments tasks into a well-defined sequence such as: plan, design, code, and test. During the planning phase a description of the system functionality is captured and a project plan is put forth. In design, the general architecture is drawn and low level processing details are planned. Testing is performed to ensure that the system functions according to specification.

In traditional software engineering practice, this succession of tasks is executed only once during a project's duration. Known as the Waterfall Model, it is the oldest software lifecycle model, and remains widely used [3]. Yet, real projects rarely follow a sequential flow of tasks performed in a preplanned, scheduled sequence. The traditional engineering life-cycle model does not formally address the natural uncertainty that exists in most projects. Frequently, technical, scheduling, or personnel issues that arise during the development process are not easily accommodated. Finally, a working version of the software system is not built until late in the project time span. Hence, there is no early assessment of how well the product meets the user's requirements.

An alternative approach is an evolutionary software development model that cycles through the task sequence multiple times until a final product is delivered [2]. Software development becomes a series

Proceedings of the Information Visualization (IV'06)
0-7695-2602-0/06 $20.00 © 2006 IEEE

of mini-life cycles whose end product becomes starting material for the next iteration until requirements and system converge. Feedback is essential to this process. Each end product is evaluated so that revisions may be made in the system design or software components, and the project plan and deliverables may be adjusted for the next cycle stage. Hence, the software product and development plan evolve and adapt over time.

Agile software engineering techniques are built around evolutionary life cycles. Recognizing that plans change, they are adapted by the process as changes occur. Driven by customer descriptions of what is required, software is developed iteratively with a heavy emphasis on rapid, incremental delivery. Agile methods place project teams members and customers in the same workspace to promote communication and allow the team to self-organize and reorganize over time. Agile development is particularly useful for projects with unpredictable or rapidly changing requirements.

There are many agile methodologies including Extreme Programming [4], the Dynamic Systems Development Method [5], Scrum [6], Crystal [7], and Feature Driven Development [8]. The methodology developed by Jim Highsmith, called Adaptive Software Development (ASD) [9], was selected for the *Trigger* project because ASD concentrates on mission-driven planning, explicit consideration of risks, and emphasizes learning throughout the process. The aim of ASD is to provide a framework with enough guidance to prevent projects from self-destructing, but not too much to suppress creativity.

ASD provided the conceptual foundation for the management of this project to deal with the high risk factors that were anticipated for project development and the expected need to simultaneously learn and develop throughout the process. Indeed, *Trigger* was a prime candidate for agile software methods for a number of reasons. First, the members of the development team resided in New York and Los Angeles and had other commitments to their time besides the art project. Their work practices and meetings could not be precisely scheduled, thus necessitating an agile process management model. Second, the technology that would be employed for *Trigger* was new to team members, requiring it to be learned and applied on the fly. Therefore, the design and construction activities needed to co-evolve iteratively from the project's inception, in order to create the technology that would support the artwork. Finally, the project began with a general set of requirements that would change and converge over time with the system as it was developed, thus compelling the team to embrace change as the project's guiding principle.

3. Requirements Engineering

The ASD methodology helps define a project's mission. The mission sets the endpoint for the product, and all development is steered so that the mission will be accomplished. One of the most important things in defining the project mission is to discover what information is needed in order to carry out the project. This is accomplished through the process of requirements engineering. Requirements engineering is the task of capturing, structuring, and accurately representing the user's requirements so that they can be correctly embodied in systems that meet those requirements. It is the most important part of the process because without a precise understanding of the system requirements it is possible to build a well functioning system that does not perform the tasks requested by the user.

The process of capturing a user's requirements usually encompasses interviews, observations, and analysis of current and previously built systems. The end result is a set of functional requirements that specify what is to be done without indicating how to do it.

Requirements engineering for this project began when the computer scientists met the artist, Jody Zellen, for a first interview. Zellen made a preliminary visit to the Digital Gallery in late Fall 2004 in preparation for her artist-in-residence in order to assess the space and discuss possible projects for the installation. She expressed an interest in building a projector-based interactive installation in which a viewer could control what is seen and heard. The work would saturate the gallery space with dynamic video and sound under viewer control. She was experienced with multi-projector installations. In particular, her work entitled *Disembodied Voices* (2004) was a five projector multi-sensory interactive installation in which the viewer moved through a series of experiences exploring the differences between public and private life, and the global phenomenon of cell phone conversations which have become an ubiquitous irritant in contemporary society. In this installation five dynamic animations filled the walls of the gallery space - four of which were looping DVD projections that depicted anonymous individuals moving through a series of public spaces. The fifth projection was connected to a computer that contained the interactive component of the installation - a series of 'toy' cell-

phones that hung from the gallery's ceiling. When visitors pressed one of the buttons on the ten phones an event was triggered that changed the representations projected.

Discussions began with the possibility of expanding this technology, but soon progressed into the use of sensors. It was pointed out that the hardwired interface of the previous work could not be easily expanded or adapted, but a more general solution of using sensors would allow capture of viewers' movements throughout the space. These movements could be used to trigger events that changed the projected imagery and sounds. Further discussions focused on the possible number of sensors, projectors, types of computers, and communications among them. In the end, a set of functional requirements for the system was agreed upon for the artwork:

- A system would be built that would support a large number of projectors, sensors, and speakers.
- The system's sensors would capture viewer motion.
- Captured motions would trigger events
- Events would be communicated to the artist's multimedia programs.
- The multimedia programs would change the content of the displays and alter sounds.

It was recognized that these requirements would evolve with time as the system progressed towards completion.

4. Risk Assessment

Risk assessment begins early in the development process to identify, isolate, and control those project risks that could hinder or prevent the successful, timely delivery of the system. Any component of the system or member of the project could pose a risk. For example, although jointly all participants exhibited a breadth of computer experience, there were notable lapses. The artist was familiar with programming Macromedia Flash; the computer scientists were not. The artist used Macs; the computer scientists used PCs. The computer scientists were familiar with most other languages such as C/C++, Java, Basic, etc.; the artist was not. Everyone was experienced with building web applications. No one had experience with sensors, but the computer scientists had some experience with digital electronics. In order to mitigate these risks, early decisions needed to be made about hardware, software, and project responsibilities.

The sensor system was perceived as the greatest risk, because no one had any knowledge of sensor technology. The first issue was to decide what technology could be used to support sensors.

One approach was to attach sensors to a midi controller. Midi ports are usually available on a PC's audio card. What would be needed was an interface that would take a sensor's analog signal and send it to the PC's midi connector. Any software that accessed the midi port could then monitor the signal. Software such as Max/MSP from Cycling74 could do this [10]. It is a robust program for interfacing midi devices to computers and is used extensively in the performing arts. It uses a visual programming paradigm similar to scientific visualization systems such as OpenDX [11] and AVS [12], but it is expensive and node locked. Hence a license would have to be purchased for each machine in the installation. In addition, there were no interfaces between Flash and midi controllers at the time the project began; only Macromedia Director could access the midi port, and no one had experience with Director.

Another approach was to use microcontrollers to interact with the sensors. Microcontrollers are self-contained computers that possess multiple I/O channels to control lights, motors, and sensors. They can be programmed in languages such as Basic, Java, and C. Once programmed, microcontrollers can run unaided or communicate with a PC through a standard serial or USB port. Microcontrollers are inexpensive and widely used in computer education and robotics research. In addition, there is a wealth of Web resources supporting microcontroller education and applications.

The final decision was to use a microcontroller because it could be easily programmed and was supported by technical resources on the Web. The implication of this decision was that the computer scientists would take initial responsibility for the design and building of the system module to ensure that this component could be delivered on time.

Parts of the software system were sources of risk. In particular, the artist's animations needed to be interfaced with the microcontroller through the serial port. This could not be done directly with Flash. Instead of building a new system from the scratch in a language that could access the serial port and play Flash animations, it was decided to build an interface controller that was an intermediary between the Flash program and serial port. The result of this decision was that the artist became responsible for creating an interface between her program and the controller, and the computer scientists became responsible for building the controller and interface to the microcontroller. This

mitigated development risks by making each team member responsible for his or her own component.

Finally, because the artwork was site-specific, final acceptance testing could not occur until the entire system was in place. Acceptance testing is conducted to determine whether a system satisfies its requirements so the customer can determine whether or not to accept the system. This constituted a risk because testing of the complete system could not be done until the last minutes before the show was to open. Hence, any system-wide design flaws would not be found until installation. Testing each of the three computer systems individually before installation mitigated this risk because these systems act independently.

5. Managing the Development Process

Time was the most important constraint on the project's lifecycle. Since Zellen resided in California, and had professional commitments there, it would be necessary to adapt to her schedule. Josh Rose, the undergraduate computer science student working on the project, was constrained by his academic schedule commitments during the semester and enforced holidays such as the month-long January break and summer recess when he returned to his out-of-town home. Marchese had commitments to teaching, research, managing CAM and the Digital Gallery.

Zellen traveled to New York in the late autumn of 2004 to discuss the project's requirements. In keeping with the agile process, all the participants were brought together during the artist's twelve-week residency in New York during spring 2005. She returned again for a week in autumn 2005 to mount the work, which was installed as a one-person show.

Most of the technological decisions had to be made before Zellen's twelve-week residency so the computer scientists would have had some experience with the sensor systems and then could design and build the application. This set up the first milestone, scheduled for the end of February 2005.

The artist's residency ended by summer vacation, a time when project team members would have other commitments and needed to work apart. Thus, the beginning of summer break created the second milestone at which a functioning prototype needed to be demonstrated so all team members could feel secure that the exhibition could be scheduled for autumn 2005. The artwork's opening constituted the third milestone for delivery of the complete system.

Each one of these milestones constituted strict deadlines that had to be met for the project to be a success.

6. Architecture and Design

The system architecture defines the high level interrelationships among components without specifying the processing details. *Trigger* possessed a simple architecture of three components: microcontroller-sensor system, application software, and interface software between sensors and application.

Of the three modules, the microcontroller-sensor system was the most volatile component exhibiting the greatest amount of change. The initial system requirements called for three computers, each computer system controlling two projectors and four distance sensors. The sensors would be used in tandem to detect the direction of viewers' movements and use their proximity to the sensor to trigger a variety of events. During the course of development the system design was updated to reflect experiments with different types of sensors (e.g. infrared vs. sonar) and assessments of their placement in the gallery space. The gallery space was too small to accommodate twelve sensors, and the synchronizing of multiple sensors proved too time consuming given the project's time constraints and the inability to fine tune them until the final installation. Ultimately, the sensor system was simplified to two sonar sensors per computer, each controlling one projector.

The interface controller component was responsible for reading sensor data from the microcontroller-sensor module, interpreting it, and issuing key pressed events that could be recognized by the Flash application. Simply put, when a viewer crossed a sensor's path, the interface controller would register the dramatic change in the sensor's readings and issue an appropriate key command that, when recognized by the Flash application, would change the video displayed. The main implementation challenge was timing. The software needed to recognize the difference between sensor interference and a true sensor event. It needed to sort out multiple triggers by the same individual standing in the ultrasonic beam from the passage of a sequence of viewers. In the end, a timing algorithm and initial constraints derived through experimentation were built into the system with knowledge that they would be adjusted at installation.

The artist's Flash application was the most stable component. The core design would not differ from Zellen's previous work, *Disembodied Voices*, which

took keyboard events as input to trigger changes in video. However, it needed to be expanded to two displays and accommodate a greater number of state changes and timings.

7. System Integration and Testing

Trigger opened on October 28th, 2005. (Figures 1 - 3 display snapshots of the installation in the Digital Gallery's atrium stairwell.) The six days leading up to Trigger's opening were allotted for installation. The artist arrived with a general sense of the appropriate placement of the projectors on the mezzanine, second, and third floors, based on previous analyses of the space. The computer scientists had done the same for sensor arrangement. Projector placement occured first, followed by sensor and speaker placement. Once the six projectors were set and three computers stationed, sensors were put in place and wired to the microcontrollers. Integration testing began at this juncture, ending on opening day. The developers systematically walked through the space triggering all video and sound sequences. All systems worked as designed subject to the following two expectations: the sensors needed to be recalibrated for the small space in order to minimize interference effects, and timings between triggered events were adjusted to take into consideration viewer's movements. Multiple walk-throughs of the installation by the artist before the opening constituted the final acceptance test of the system.

Conclusions

In order to support collaboration between artist and scientists for the creation of a digital work of art, it was necessary to address issues involving project communication, scheduling and deadlines, cost, staffing, project monitoring, and product construction. By setting the development within the context of agile software engineering methodologies, it was possible to structure the development process and guide the product toward timely completion. The ASD method is particularly useful because it minimizes the management infrastructure and focuses on communication to foster collaborative problem solving. In conclusion, it was found that agile methods are useful in the management of artistic collaborations where there are diverse risks involving technology, time, budgets, and people.

Acknowledgements

Pace University's Center for Advanced Media's artist-in-residence was made possible through the support of Dr. Susan Merritt, Dean, Ivan G. Seidenberg School of Computer Science and Information Systems.

References

[1] J. Zellen, *Trigger,* Pace Digital Gallery, [cited Apr. 16, 2006], available from World Wide Web: <http://csis.pace.edu/digitalgallery/zellen/Trigger.pdf>.

[2] R.S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, New York, NY, McGraw-Hill, 2005.

[3] P.A. Laplante and C.J. Neill, "'The demise of the waterfall model is imminent' and other urban myths," *ACM Queue*, vol. 1, no. 10, Feb. 2004, pp. 10-15.

[4] K. Beck, *Extreme Programming Explained: Embrace Change*, Addison-Wesley Professional; Boston, MA, 1999.

[5] DSDM Consortium and J. Stapleton, *DSDM: Business Focused Development*, 2nd Edition, Upper Saddle River, NJ, Pearson Education; 2003.

[6] L. Rising and N.S. Janoff, "The scrum software development process for small teams," *IEEE Software*, vol. 17, no. 4, July /August 2000, pp. 2-8.

[7] A. Cockburn, *Crystal Clear: A Human-Powered Methodology for Small Teams*, Boston, MA, Addison-Wesley Professional, 2004.

[8] S.R. Palmer and J. M. Felsing, *A Practical Guide to Feature-Driven Development*, Upper Saddle River, NJ, Prentice Hall, 2002.

[9] J. Highsmith, *Agile Software Development: A Collaborative Approach to Managing Complex Systems*, New York, NY, Dorset House, 2000.

[10] Max/MSP, ver. 4.5, Cycling 74, Inc., San Francisco, CA.

[11] OpenDX, [cited Apr. 16, 2006], available from World Wide Web: < http://www.opendx.org/>.

[12] C. Upson, T. Faulhaber Jr., D. Kamins, D. Laidlaw, D. Schlegel, J. Vroom, R. Gurwitz, and A. van Dam, "The application visualization system: A computational environment for scientific visualization, *IEEE Computer Graphics and Applications* , vol. 9, no. 4, July 1989, pp. 30-42.

Figure 1. Composite image of *Trigger* taken from mezzanine level.

Figure 2. Image of *Trigger* showing overlapping video.

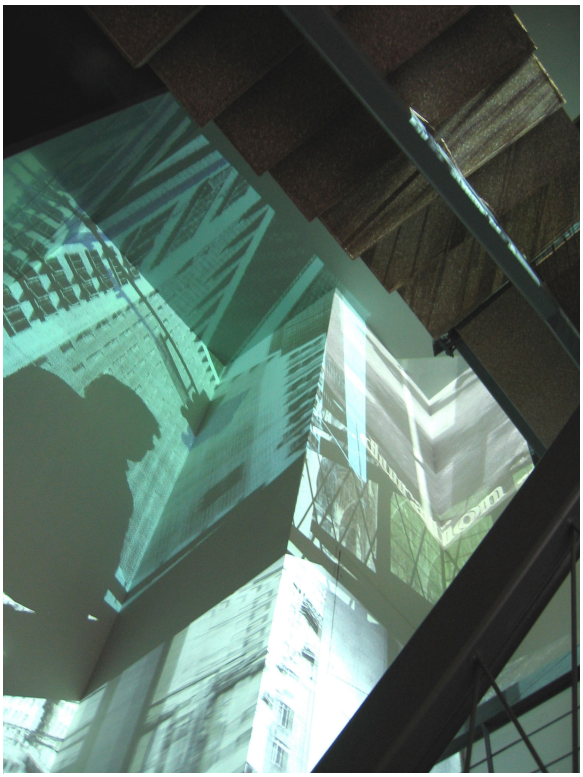

Figure 3. *Trigger* as viewed from the ground floor.