Arcade Game Maker Product Line

Architecture Evaluation Report

John D. McGregor

February 2004

Table of Contents

Executive Summary

This report presents the results of an architecture evaluation of the Arcade Game Maker Product Line software architecture, which took place at Clemson SC, on February 5, 2004. This evaluation was performed by Luminary Software and followed the Architecture Tradeoff Analysis Method[SM] (ATAM[SM]) evaluation process.

The ArcadeGame Maker Product Line architecture was evaluated to identify the basic risks associated with developing the defined products using the proposed architecture.

A summary of the results of the evaluation are:

- The architecture is appropriate for the current scope of the product line.
- The architecture is sufficiently well structured to support an upgrade to more complex games.

Two risk themes were identified:

1. The ability of the architecture to accommodate future games is questionable. The architecture is very simple in order to keep costs low. As a result certain techniques such as dividing the graphics field into regions to speed the search for collisions have not been used. Hence the ability of the architecture to support a large number of objects on the screen and still meet its performance goals is uncertain. The modified Model-View-Controller architectural pattern will not easily support the addition of more windows to display different information.
2. The realism of the games is questionable. The architecture does not support the sophisticated input and output devices that make interaction more natural. The graphics are very simple to allow the games to run on almost any hardware.

1. Introduction

This report presents the results of an architecture evaluation of the Arcade Game Maker Product Line software architecture, which took place at Clemson SC, on February 2, 2004. This evaluation was performed by the Luminary Software and followed the Architecture Tradeoff Analysis Method[SM] (ATAM[SM]), a method for evaluating a software system's architectural decisions in light of desired system quality attributes.

Evaluations using the ATAM take place in two main phases. In Phase 1 the evaluation team interacts with the architect and a few other key stakeholders (such as a project manager or customer/marketing representative). The evaluation team and the system stakeholders will walk through all of the steps of the ATAM, gathering information about the system, its important quality attributes, and its architecture. We begin analyzing architectural decisions in light of the quality attributes, uncovering risks and tradeoffs. The evaluation team will continue the analysis after the Phase 1 meeting, interacting with the architect(s) as necessary to elicit the necessary information. This interaction typically takes several weeks.

In Phase 2, the evaluation team walks through all steps of the ATAM with a larger set of system stakeholders and the work from Phase 1 is reviewed with the larger group of stakeholders. With this larger group of stakeholders, important quality attributes are illuminated, and analysis of the architecture's ability to support those goals continues.

Due to the simplicity of the Arcade Game Maker Product Line architecture Phase 1 and Phase 2 were combined. There was no non-technical stakeholders to involve at the latter phase.

The system stakeholders (architects, managers, developers, testers, integrators, etc.) participating in the Arcade Game Maker Product Line ATAM evaluation are shown in Table 1 below.

Table 1. Attendees in the Arcade Game Maker Product Line **Evaluation**

Name	Organization	E-mail	Role

The ATAM evaluation team members, and their assigned roles in the <system> ATAM evaluation are shown in Table 2 below.

Table 2. Evaluation Team for the Arcade Game Maker Product Line **Evaluation**

Name	Organization	E-mail	Role

The remainder of the report is organized as follows:

Section 2: The Architecture Tradeoff Analysis Method (ATAM)
Section 3: Business Drivers Presentation
Section 4: Architecture Presentation
Section 5: Untility Tree
Section 6: Scenario Generation/Prioritization
Section 7: Analysis of Architectural Approaches
Section 8: Risks, Sensitivities, and Tradeoffs
Section 9: Conclusions and Next Steps

2. The Architecture Tradeoff Analysis Method (ATAM)

The ATAM relies on the fact that an architecture is suitable (or not suitable) only in the context of specific quality attributes that it must impart to the system. The ATAM uses stakeholder perspectives to produce a collection of scenarios that define the qualities of interest for the particular system under consideration. Scenarios give specific instances of usage, performance requirements, growth requirements, various types of failures, various possible threats, and various likely modifications. Once the important quality attributes are identified in detail, then the architectural decisions relevant to each one can be illuminated and analyzed with respect to their appropriateness.

The steps of the ATAM are carried out in two phases. In the first phase, the evaluation team interacts with the system's primary decision-makers: the architect(s), manager(s), and perhaps a marketing or customer representative. During the second phase, a larger group of stakeholders is assembled, including developers, testers, maintainers, adminstrators, and users. The two-phase approach insures that the analysis is based on a broad and appropriate range of perspectives.

Phase 1:

1. **Present the ATAM.** The evaluators explain the method so that those who will be involved in the evaluation have an understanding of the ATAM process.
2. **Present business drivers.** Appropriate system representative(s) present an overview of the system, its requirements, business goals, context, and the architectural quality drivers.
3. **Present architecture.** The system or software architect (or another lead technical person) presents the architecture.
4. **Catalog architectural approaches.** The system or software architect presents general architectural approaches to achieve specific qualities. The evaluation team captures a list, and adds to it any approaches they saw during Step 3 or learned during their pre-exercise review of the architecture documentation. For example, "a cyclic executive is used to ensure real time performance." Known architectural approaches have known quality attribute properties, and these will help carry out the analysis steps.
5. **Generate quality attribute utility tree.** Participants build a utility tree, which is a prioritized set of detailed statements about what quality attributes are most important for the architecture to carry (such as performance, modifiability, reliability, or security) and specific scenarios that express these attributes..
6. **Analyze architectural approaches.** The evaluators and the architect(s) map the utility tree scenarios to the architecture to see how it responds to each important scenario.

Phase 2:

Phase 2 begins with an encore of the Step 1 ATAM presentation and a re-cap of the results of steps 2 through 6 for the larger group of stakeholders. Then:

7. **Brainstorm and prioritize scenarios.** The stakeholders brainstorm additional scenarios that express specific quality concerns. After brainstorming, the group chooses the most important ones using a facilitated voting process.
8. **Analyze architectural approaches.** As in Step 6, the evaluators and the architect(s) map the high priority brainstormed scenarios to the architecture.
9. **Present results.** A presentation and final report are produced that capture the results of the process and summarize the key findings.

Scenario analysis produces the following results:

- a collection of sensitivity and tradeoff points. A sensitivity point is an architectural decision that affects the achievement of a particular quality. A tradeoff point is an architectural decision that affects more than one quality attribute (possibly in opposite ways).

- a collection of risks and non-risks. A risk is an architectural decision that is problematic in light of the quality attributes that it affects. A non-risk is an architectural decision that is appropriate in the context of the quality attributes that it affects.
- a list of issues, or decisions not yet made. Often during an evaluation, issues not directly related to the architecture arise. These may have to do with an organization's processes, personnel, or other special circumstances. The ATAM process records these so that they may be addressed by other means. The list of decisions not yet made arises from the stage of the life cycle of the evaluation. An architecture represents a collection of decisions. Not all relevent decisions may have been made at the time of the evaluation, even when designing the architecture. Some of these decisions are known to the development team as having not been made and are on a list for further consideration. Others are news to the development team and stakeholders.

Results of the overall exercise also include the summary of the business drivers, the architecture, the utility tree, and the analysis of each chosen scenario. All of these results are recorded visibly so that all stakeholders can verify they have been correctly identified.

The number of scenarios that are analyzed during the evaluation is controlled by the amount of time allowed for the evaluation, but the process insures that the most important ones are addressed

After the evaluation, the evaluators write a report documenting the evaluation and recording the information discovered. This report will also document the framework for on-going analysis discovered by the evaluators. This is the report in your hands. A detailed description of the ATAM process can be found in [6] and [7].

3. Business Drivers Presentation

John D. McGregor described Arcade Game Maker's business objectives for the Arcade Game Maker Product Line and the driving business requirements for the Arcade Game Maker Product Line. Products and the architecture goals derived from these requirements included these points:

- The complete description of driving business requirements can be found in the Business Case for the Arcade Game Maker Product Line document which can be located on the product line document map.

- The primary driving requirement is to maintain low cost. Many of the products are intended either for Arcade Game Maker to give away via the web or for customers to give aways at conventions.

- A secondary driving requirement is to offer the user an interesting game playing experience. People who play games are easily frustrated and quick to scorn games that they believe to be inferior. The game must be sufficiently fast and realistic to engage a game player.

4. Architecture Presentation

John D. McGregor of Arcade Game Maker next presented the architecture of the Arcade Game Maker Product Line.

Several architectural approaches were identified. These include:

- A complete Model-View-Controller (MVC) approach
- A modified MVC approach
- An approach that uses an encapsulated game engine

The modified MVC approach was chosen for the product line. The traditional MVC approach required too much overhead communication to be used in a game situation. The game engine approach was too complex for the class of games being developed.

The presentation focuses on the chosen architecture.

The compete Arcade Game Maker Product Line architecture can be found in the product line's document map. Here we present a couple of the views.

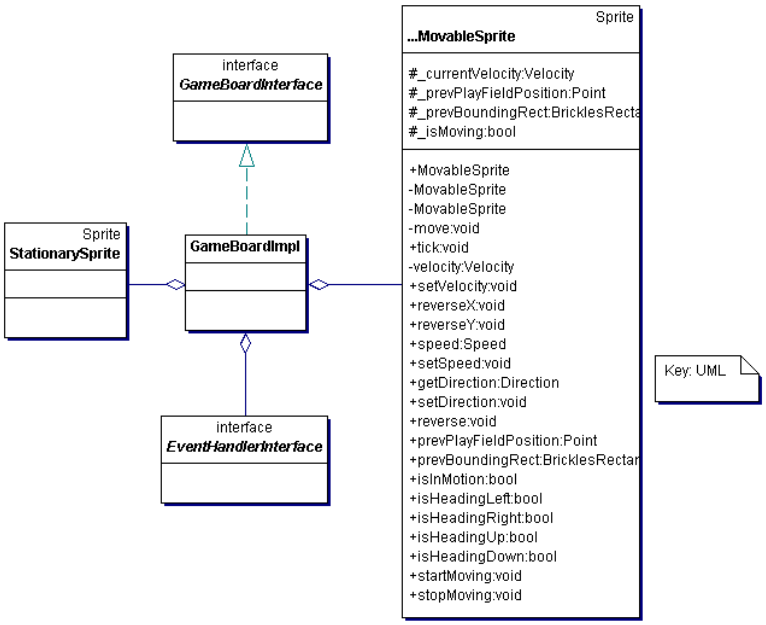

Figure 1 - GameBoard-centric View

The GameBoard is the central feature of the architecture. It is a container that contains all of the action of the game. The StationarySprites are the obstacles into which the MovableSprites collide. Movement and collision are the two major actions that occur in the games.

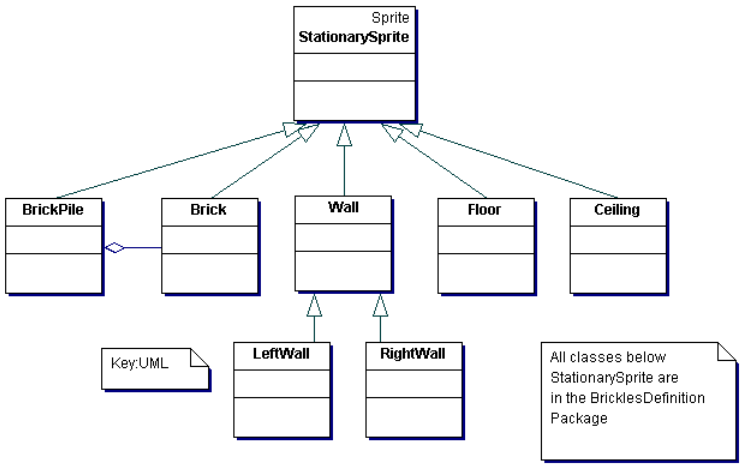

Figure 2 - The Specialization Hierarchy for StationarySprites

This view illustrates how obstacles are defined.

5. Utility Tree

The utility tree provides a vehicle for translating the quality attribute goals articulated in the business drivers presentation to "testable" quality attribute scenarios. The tree contains Utility as the root node. This is an expression of the overall "goodness" of the system. In the case of *Arcade Game Maker Product Line* the second level nodes were *performance and modifiability.*

Under each of these quality attributes are specific concerns. These concerns arise from considering the quality-attribute specific stimuli and responses that the architecture must address. For example, in the case of *Arcade Game Maker Product Line, performance* was broken into *two concerns: the total number of Sprites on the screen and how many Sprites have to be checked for collisions after each tick of the game clock*.

Finally, these concerns are represented by a small number of scenarios. These scenarios are leaves of utility tree and the utility tree thus has four levels.

A scenario represents a use of, or modification of the architecture, applied not only to determine if the architecture meets a functional requirement, but also (and more significantly) for prediction of system qualities such as performance, reliability, modifiability, and so forth.

The scenarios at the leaves of the utility tree are prioritized along two dimensions: [importance to the system, perceived risk in achieving this goal]. These nodes are prioritized relative to each other using relative rankings such as High, Medium, and Low.

Phase I: Quality Attribute Utility Tree		
Quality Attribute	Performance	
Attribute Concerns	A. The growth rate of the number of stationarySprites may degrade the performance of the game.	
Scenarios	1. With each tick of the simulation clock every stationarySprite is checked to determine whether it has been hit by one of the MovableSprites. The next tick cannot be handled until all current sprites have been checked for collisions.	(H,H)
	2. All stationarySprites are created at the start of a new match for a game. As the number of sprites grows, the startup time for a game will grow longer. This may become unacceptable.	(H,H)
	3.	
Attribute Concerns	B. The rate of ticks generated by the timer may be too slow for smooth animation.	
Scenarios	1. The game player sees the animation as jerky motion.	(H, L)
	2.	
	3.	
Attribute Concerns	C. The rate of ticks generated by the timer may be too fast for the user to react..	
Scenarios	1. The game player can not move the paddle sufficiently fast to intercept the puck.	(H, L)
	2. :	**:**
	3. :	(I, D)

Phase I: Quality Attribute Utility Tree

Quality Attribute	Modifiability	
Attribute Concerns	A. The range of scoring procedures for the game sin the product line may be too broad to fit within the architectural approach.	
Scenarios	4. Pong and Brickles use simple counting schemes while Bowling requires more complex logic to compute the score.	(H,H)
	5.	
	6.	
Attribute Concerns	B. The modified MVC approach we are using may not be sufficiently flexible to accommodate later additions to the product line.	
Scenarios	4. The addition of pinball may require so many more display attributes that a full blown MVC would be a better choice.	(H, H)
	5.	
	6.	

6. Scenario Generation/Prioritization

In addition to the scenarios at the leaves of the utility tree, a scenario elicitation process allows stakeholders to *contribute* additional scenarios that reflect their concerns and understanding of how the architecture will accommodate their needs. A particular scenario may, in fact, have implications for many stakeholders: for a modification, one stakeholder may be concerned with the difficulty of a change and its performance impact, while another may be interested in how the change will affect integrability of the architecture. Table 3 shows the scenarios that were collected by a round-robin brainstorming activity. After the scenarios were generated they were prioritized using a voting process in which participants were given 5 votes, which they could allocate to any scenario or scenarios they chose. The number of votes each scenario received is shown in the right most column of Table 3.

Table 3. Brainstormed Arcade Game Maker Product Line Scenarios

Phase 2: Brainstormed Scenarios		
Scenario Number	**Scenario Text**	**Number of Votes**
1.	A person downloads the executible of one of the games to their computer. When they attempt to use the game, they do not have sufficient memory. The attempt to use the game causes their operating system to reboot. Their impression is that our game caused the problem. We lose their goodwill.	5
2.	A person downloads the executible of one of the games to their computer. After playing the game for some time, the program crashes and they lose their progress toward the goal. We lose their goodwill.	5
3.	A person downloads the executible of one of the games to their computer. When they attempt to save their score the program erases their hard drive. They file suit seeking damages for the replacement of their data.	5

7. Analysis of Architectural Approaches

Six scenarios from the Phase 1 utility tree and the Phase 2 scenario generation/prioritization process were examined in detail vis-à-vis the *AGM* architecture. The scenarios that were examined are listed in the previous tables.

The use of the game engine would reduce the risks posed by the elicited scenarios. The game engine has proven functionality to handle these actions. The tradeoff is that it would take a long time for the AGM developers to learn the game engine and to use it properly.

The MVC approach requires that the Model accept generic Views and to notify all of them whenever there is a change in the Model, which would be every clock tick in our products. The Views then all update themselves regardless of whether the data they are presenting is changed or not. Due to the small number, and relative stability, of the Views this power is unnecessary and may be harmful in terms of performance.

8. Risks, Sensitivities, Tradeoffs

The analyses of architectural approaches by applying selected scenarios and the ensuing discussions surfaced a set of risks, sensitivities, and tradeoffs.

Collected Risks

The collected risks are as follows:

risk 1: The highest risk is the potential for user dissatisfaction. The current architecture supports a very simple graphical model. The graphics are not as realistic as other available games. If users are dissatisfied, they may have a negative impression of the company and the goodwill expected from the free games will not accrue.

Non-Risks

In addition to the risks that were detected, several items were explicitly identified as being areas where there were no risks involved. These were:

non-risk theme 1.Scoring and accurately representing the games. Each of the games is sufficiently simple that the scoring is clear and easily implemented.

Collected Sensitivities

The collected sensitivities are as follows:

sensitivity 1: The number of graphical elements in a game is a sensitivity point in this architecture. If games were added to the product line that used a larger number of elements, many of the quality attributes would face degradation.

Collected Tradeoffs

The collected tradeoffs are as follows:

tradeoffs 1: There has been a tradeoff between memory use and modifiability. The full Model-View-Controller architecture results in duplication of data between the Model and the Views. Some pieces of data might be in every View plus the Model. This is not a problem for the free versions but would have been a problem for the wireless device versions. The modified MVC architecture only stores the data in one place but it is much harder to add additional Views. Because this is such a fundamental architecture element it was not made a variation.
tradeoffs 2: There has been a tradeoff between performance and simplicity. As stated previously in this report, the graphical area of the Game interface could have been subdivided into regions to improve the speed of checking for collisions. To reduce development costs, a single region was used.

Risk, Sensitivity, Tradeoff Themes

The set of risks, sensitivities, and tradeoffs presented above led to a discussion of “themes” in the architecture. These represent the key architectural issues that pose potential future problems for the success of the venture. These themes are as follows:

Theme 1 – Scalability

The ability of the architecture to accommodate future games is questionable. The architecture is very simple in order to keep costs low. As a result certain techniques such as dividing the graphics field into regions to speed the search for collisions have not been used. Hence the ability of the architecture to support a large number of objects on the screen and still meet its performance goals is uncertain. The modified Model-View-Controller architectural pattern will not easily support the addition of more windows to display different information.

Theme 2 – Realism

The realism of the games is questionable. The architecture does not support the sophisticated input and output devices that make interaction more natural. The graphics are very simple to allow the games to run on almost any hardware.

9. Conclusions and Next Steps

The results of the ATAM showed the need to modify the existing architecture to include user and system models that allow the user to control more aspects of the game. The detailed design process will consider whether to do this through menu selections or through a configuration file. The next step is to revise the architecture to accommodate these models.

The performance of the current design is adequate but just barely. We will not do anything explicitly to improve performance but the above described work must not degrade performance at all.

If this product line proves successful we will revisit the architecture with a view to modify it to support a second product line.

10. References

[1] L. Bass, P. Clements, R. Kazman, *Software Architecture in Practice*, Addison-Wesley, 1998.

[2] R. Kazman, S. J. Carriere, "Playing Detective: Reconstructing Software Architecture from Available Evidence", *Automated Software Engineering,* 6:2, April 1999.

[3] R. Kazman, M. Barbacci, M. Klein, S. J. Carriere, "Experience with Performing Architecture Tradeoff Analysis", *Proceedings of ICSE 21*, (Los Angeles, CA), May 1999.

[4] M. Klein, R. Kazman, "Attribute-Based Architectural Styles", CMU/SEI-99-TR-22, Software Engineering Institute, Carnegie Mellon University, 1999.

[5] P. Kruchten, "The 4+1 View Model of Software Architecture", *IEEE Software*, Nov. 1995, 42- 50.

[6] P. Clements, R. Kazman, M. Klein, *Evaluating Software Architectures*, Addison-Wesley, 2002.

[7] R. Kazman, M. Klein, P. Clements, "ATAM: Method for Architecture Evaluation", CMU/SEI-2000-TR-004, Software Engineering Institute, Carnegie Mellon University, 1999.