iTiger: Architecture Documentation, Volume 1 - Beyond Views

John D. McGregor

March 2008

Contents

List of Figures

List of Tables

Revision Control Table				
Version Number	**Date Revised**	**Revision Type A-Add, D-Delete, M-Modify**	**Description of Change**	**Person Responsible**
1.0	3/08	A	Created new document	JDMcGregor

1 Overview

The iTiger project is intended to provide unique capabilities to attendees at Clemson University home football games. iTiger provides the ability to see replays during the game on handheld devices, to interface with social networks, and to retrieve archived game video.

Long term the software that forms the basis for iTiger may be made available to other organizations.

2 iTiger System Overview

2.1 History

During the 2007 – 2008 academic year networking researchers began experimenting with wireless networking hardware and software. At the start of the Spring 2008 semester the course CpSc 875, software architecture, took on iTiger as the project for the semester.

2.2 Description

The iTiger system will provide an in-the-stadium experience that enhances the other traditional features of a home football game. Using wireless technology, iTiger will provide attendees with access to the same plays as seen on the stadium display as well as special features such as access to their social networking accounts. iTiger users can communicate with each other during the game and otherwise.

3 iTiger Architecture View Packet Template

We adopted the view packet template presented by Clements and colleagues [Clements 03]. In this section, we briefly describe each part of it. The template is used in the *iTiger: Architecture Documentation, Volume 2 - Software Architecture Views* to organize each view packet.

3.1 View Packet Template Description

3.1.1 Primary Presentation

This presentation provides the basic model for this part of the architecture.

3.1.2 Element Catalog

The element catalog includes a description of each element included in a view. The items that are "elements" in a particular view are the atomic items that characterize that view. For example, in the module view, the elements are modules.

Properties of the elements. The element's properties determine its impact on the quality attributes. For example, an element may have a large negative impact on the performance of the overall system.

Relations and their properties. The element's relations determine how they are associated with each other. For example, an element may aggregate or be a specialization of another element.

Element interfaces. The element's interface shows its publicly available services using method signatures.

Element behavior. AADL and UML diagrams show the element's dynamic behavior. For example, behavior is shown with UML sequence and activity diagrams.

3.1.3 Context Diagram

This presentation places the elements contained in a view packet in the overall architecture.

3.1.4 Variation Guide

This section describes the variations for each element included in a view.

3.1.5 Architecture Background

This section describes architectural decisions.

Rationale. The rationale documents decisions that resulted in the architecture shown in the view.

Analysis results. The analysis results provide the data that back up the decisions.

Assumptions. The assumptions include data that underlie the decisions.

3.1.6 Other Information

This section includes miscellaneous information that didn't fit into another section.

3.1.7 Related View Packets

This section describes the following types of related view packets:

- ones that surround the current view packet at the same level of detail
- ones that include the current view packet at a higher level
- ones that describe additional elements with the current view packet

4 Mapping Between Views

In general, we use AADL to document the static architecture. In AADL, there are static diagrams that describe definitional units, such as systems and processes, and dynamic elements of these diagrams that describe operational activities such as flows and connections. Any operational unit that is used must correspond to a definitional unit. Therefore, there is a general mapping from dynamic diagrams to static diagrams.

We augment the AADL diagrams with SysML diagrams that provide a means to seamlessly move between hardware and software.

5 Rationale, Background, and Design Constraints

In this section, we document several architecture decisions that resulted in the current architecture.

5.1 User interaction

Our initial concept was to use an implementation of the model-view-controller (MVC) architecture to be the main vehicle for user interaction. The requirements to play videos and interface with social networking web site made us think more about the scope of user interaction. The use of an already built internet browser seemed an attractive alternative.

5.1.1 Model-View-Controller (MVC)

Controllers provide input to the system. The input is routed to the view or the model as appropriate. In iTiger, the keyboard and mouse serve as controllers.

The views present information to the player in a variety of forms. In iTiger, the graphical interface has several fixed items and a video player showing changing content. The mouse controller allows the user to select menu items and to use a drag and drop interface. The view has the responsibility to ask the model for the data it needs to build its presentation.

The model is the computation engine for the system. The model would maintain the status of the display. The model is obligated to inform all views when its state has changed. In iTiger a primary state change would change the menus available to the user.

The MVC pattern focuses on adding an arbitrary number of views to a given model. Tabular and graphical views of a data table might be useful, for example. The tradeoff is that a view updates itself by querying the model for the information it needs after the model has notified the view of a state change. The update operation is shown in a sequence diagram in Figure 1.

iTiger needs a number of different views. To remain as flexible as possible with respect to the hardware form factor, we would like to simply paint the device service in a view so that buttons, windows and all other elements are programmable. This can be done with a number of pre-selected views.

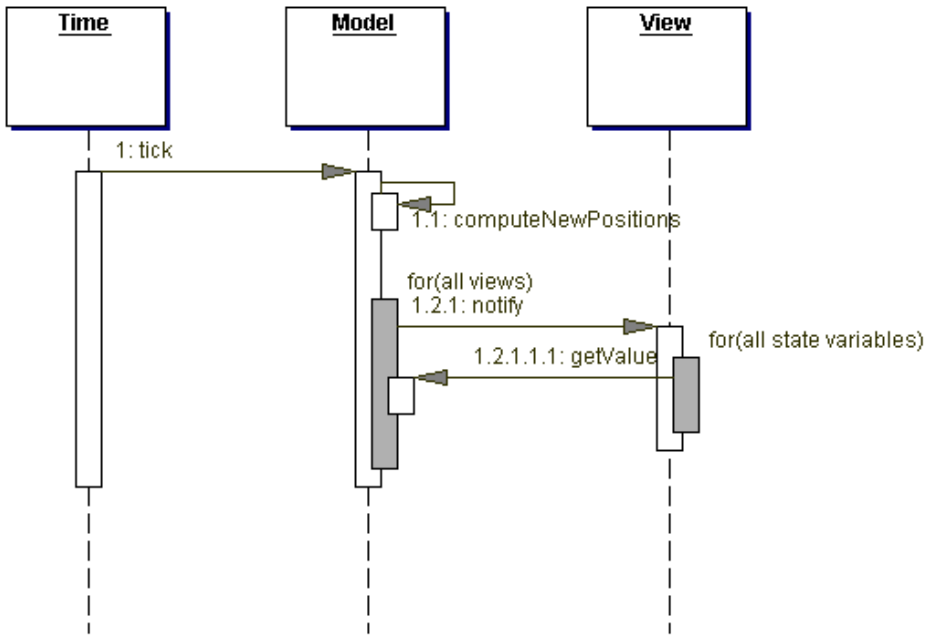


Figure 1: Model-View-Controller (MVC) Update Operation

5.1.2 An Alternative

An alternative is to use an internet browser and consider it a black box in terms of the iTiger architecture. The browser alternative also determines the connection protocol between the presentation and client tiers. It also imposes a number of requirements on the presentation tier.

This architecture is compared to the MVC architecture in Table 1.

5.1.3 Analysis

The advantages and disadvantages of each design are shown in Table 1.

Table 1: Comparison of MVC and Monolithic Approaches

	Advantages	**Disadvantages**
MVC	The user interface can be made to be exactly what we wish it to be.	Requires programming expertise; cannot be modified dynamically
Internet browser	The user interface is very modifiable with easy to use scripting languages	Scripting languages are not as flexible as general programming languages

After reviewing the table, the team chose to use the browser version of the architecture. In particular, they rated performance as a high priority and the browser is so widely used that its implementation has been iteratively improved.

5.2

The requirement for streaming video initiated a study of options.

5.2.1 Include streaming video in web server

The Apache web server has the possibility of serving video. However, it does so through the same channel as it serves http data. This results in less than optimal throughput.

5.2.2 The Alternative – a specialized video server

A specialized video server uses a different connection protocol than that used by a generic web server. This approach accounts for long lived transactions as opposed to the packet based approach of TCP/IP.

5.2.3 Analysis

The advantages and disadvantages of each design are shown in Table 2.

Table 2: Generic web server vs video server

	Advantages	**Disadvantages**
Generic web server	Handles all actions in one product	The performance will not be optimal and may be unacceptable at times
Video server	Tuned specifically for serving video	Means a second product to maintain and possibly a second vendor to manage

The team decided to use the dedicated video server in parallel with the web server.

6 Attached Processes

6.1 Process for Modifying the Architecture

There are several reasons for architecture changes. Figure 2 describes a process that can be used for any architecture change, but we describe it using a requirements change.

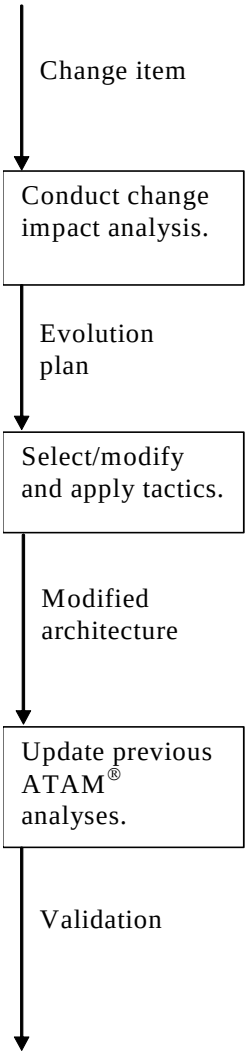
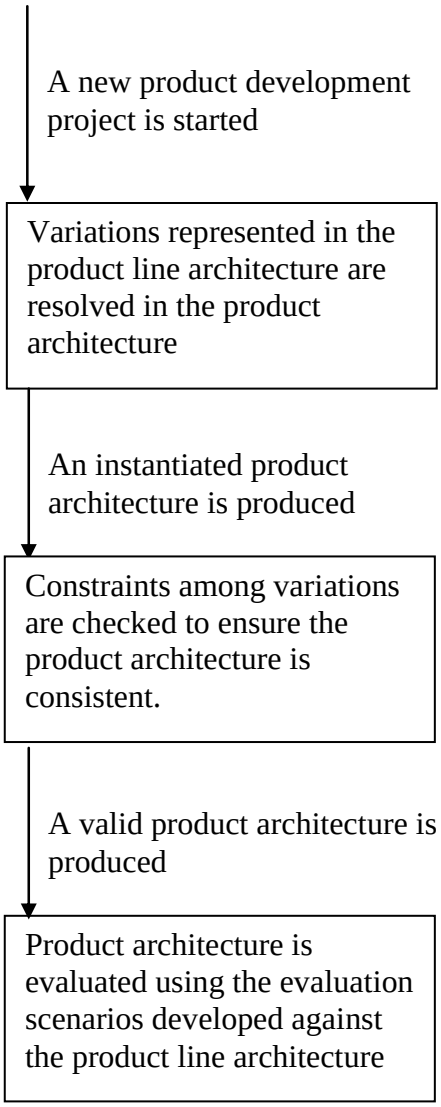


Figure 2: Attached Process for Requirements Change

When a requirement is added or modified, the impact of that change to the requirements model is determined. The architecture is reviewed to determine whether the current architecture can satisfy the new requirement. If not, existing items are examined to determine whether they can satisfy the behaviors. If not, new items are designed with responsibility for the new behavior. Any new items are placed in the same operational context as the other items.

6.2 Process for Using the Architecture to Produce a Product

This process becomes a portion of the production plan for a product.



7 References and Further Reading

For details about the references cited in this document, see *iTiger: Bibliography*.