iTiger: Architecture Documentation, Volume 2 - Software Architecture 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 document	JDMcGregor

1 Introduction

The iTiger product is a unique approach to in-stadium entertainment. The user is able to access a number of services that are unique to iTiger as well as accessing their usual web functions.

The architecture attains the qualities prescribed for it in *iTiger: Requirements Model*: that is, the product must be fast, usable, and modifiable.

This second architecture volume provides detailed models of the product's architectural model. For background information that supports these models, see *iTiger: Architecture Documentation, Volume 1 - Beyond Views*.

2 Allocation Deployment View

2.1 Allocation Deployment View Packet 1: Top level

As shown in Figure 1, we intend to deploy iTiger on several machines. The pieces are portable and the exact configuration can be modified by simply changing URLs. We show here the maximum of distribution. At this time there is a 1-to-1 relationship among the hardware platforms and the basic pieces of the software. The section on variation discusses an alternative mapping.

2.1.1 Primary Presentation

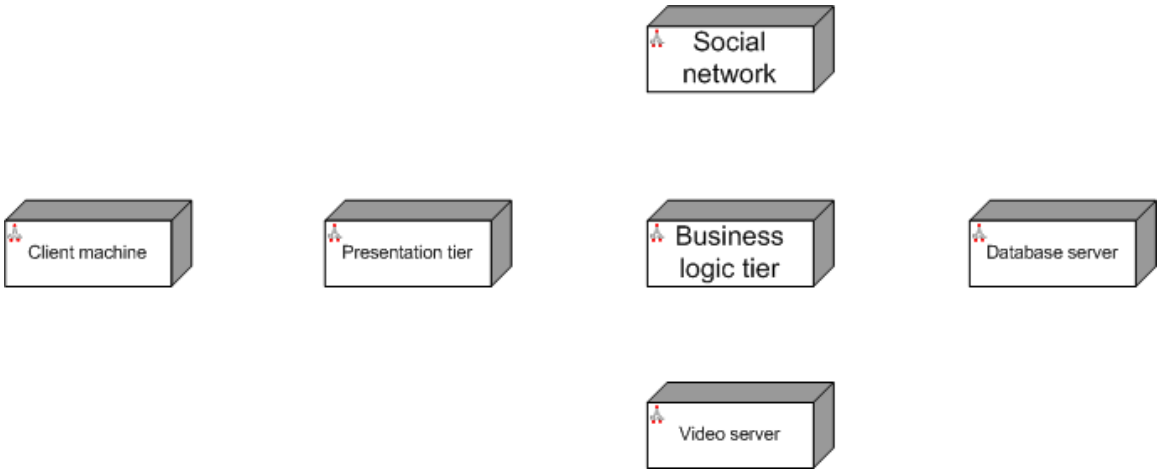

Figure 1: Process Allocation for iTiger

Each box in Figure 1is a hardware platform.

2.1.2 Element Catalog

Elements and their properties. Table 1 displays element details.

Table 1: Elements and Responsibilities for Allocation Deployment View Packet 1: iTiger

Element	Responsibilities
Client	**This is the user's entry into the system**
Presentation tier	**Does the initial handling of user queries and formats information to be displayed on the client – it is partially implemented by the web server**
Business logic tier	**Does the computation to handle a user's request**
Database tier	**Interfaces with the physical database; formats queries, does any optimization specific to the particular database being used; returns data to the business logic tier or directly to the video server**
Video server	**Special server devoted to handing the video stream**
Social networking site	**This interface allows our users access to their social network but allows us to have some knowledge of what they are doing; specifies the Open Social interface**

Relations and their properties. In this view the primary relationship is "binding". A piece of software is bound to a specific piece of hardware which determines some of its operating characteristics.

Element interfaces. The interface between the software module and the hardware it runs on will depend upon the language of implementation. For Java, the interface will be the VM.

Element behavior. The behavior of each tier will be shown more completely in the module view.

2.1.3 Context Diagram

This is the top level of the product. There is no context other than an executable running on the supported platform and a user driving the Client.

2.1.4 Variation Guide

The main variation is where each process is deployed. In particular, the presentation and business logic tiers can be sited together depending on the anticipated load on the system.

2.1.5 Architecture Background

This view was determined very early in the project.

2.1.6 Other Information

No other information applies.

2.1.7 Related View Packets

- None at the moment

3 Module Decomposition View

In this section, we describe the basic module structure of iTiger.

3.1 Module Decomposition View Packet 1: iTiger

Figure 2 shows the top level module decomposition of iTiger.

3.1.1 Primary Presentation

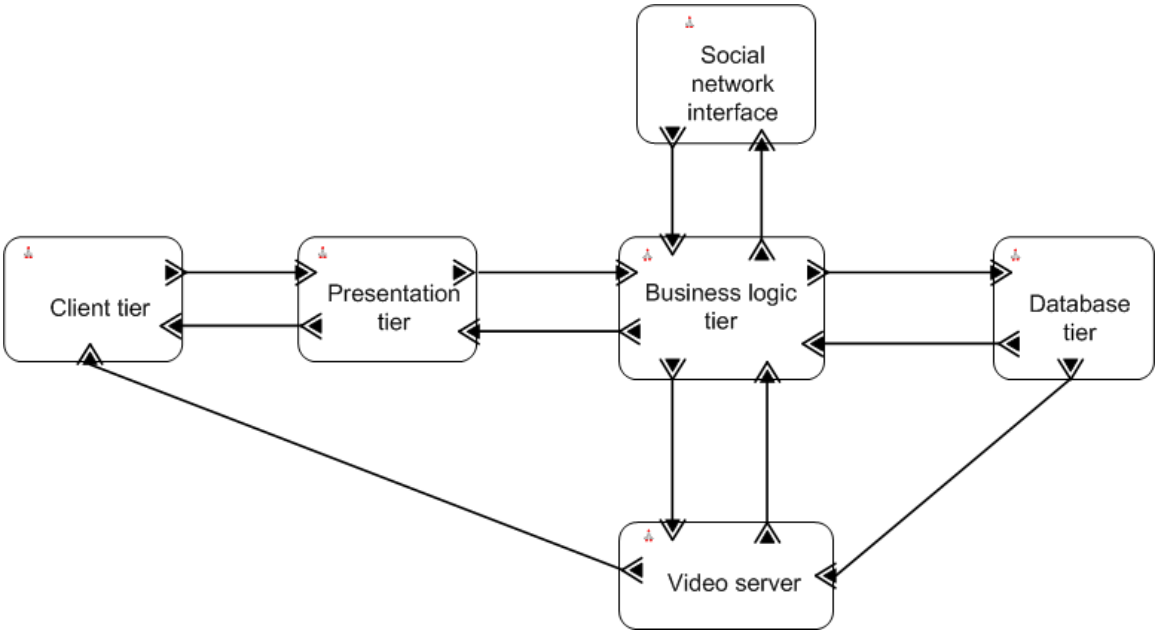

Figure 2: Module View Decomposition of iTiger

3.1.2 Element Catalog

Elements and their properties. Table 2 displays element details.

Table 2: Elements and Responsibilities for Model Decomposition View Packet 1: iTiger top level

Element	Responsibilities
Client	**This is the user's entry into the system**
Presentation tier	**Does the initial handling of user queries and formats information to be displayed on the client – it is partially implemented by the web server**
Business logic tier	**Does the computation to handle a user's request**
Database tier	**Interfaces with the physical database; formats queries, does any optimization specific to the particular database being used; returns data to the business logic tier or directly to the video server**
Video server	**Special server devoted to handing the video stream**
Social networking interface	**This interface allows our users access to their social network but allows us to have some knowledge of what they are doing; specifies the Open Social interface**

Relations and their properties.

Element interfaces.

Element behavior.

3.1.3 Context Diagram

This is the top level of the system.

3.1.4 Variation Guide

The database and social networking interfaces can each be attached to different products.

3.1.5 Architecture Background

This basic module view represents the entire set of architecture decisions. The 4 tier is a standard architecture pattern that provides maximum flexibility with respect to maintenance and use of ready-made products.

3.1.6 Other Information

No other information applies.

3.1.7 Related View Packets

Section 2.1shows the deployment of these modules to hardware.

4 Module Vendor View

4.1 Module Vendor View Packet 1: iTiger

This view shows the origin of modules by vendor (Figure 3).

4.1.1 Primary Presentation

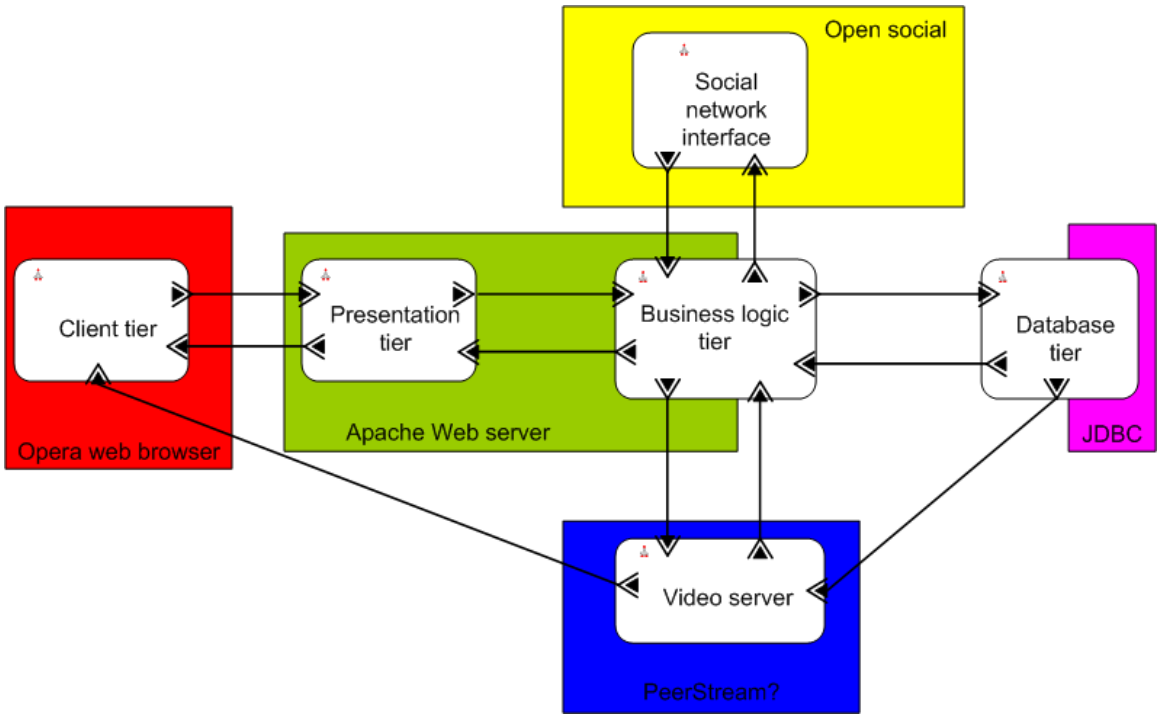

Figure 3: Game Generalization

4.1.2 Element Catalog

Elements and their properties. Table 3 displays element details.

Table 3: Elements and Responsibilities for Module Vendor View Packet 1:iTiger

Element	Responsibilities
Client	**The Opera internet browser is standard on one of the three phones we are currently targeting.**
Presentation tier	**The Apache web server handles most of the required functions.**
Business logic tier	**There are parts of this that must be created by our project**
Database tier	**This tier implements the JDBC interface. This will make it possible to swap out the actual database used.**

Video server	**No decision has been made here. The PeerServer is one possibility.**
Social networking site	**The social networking interface uses the Open Social standard.**

Relations and their properties. The relations shown in this section are “implemented by” and “acquired from.” The portion of a module falling within a colored box is implemented by the product represented by the box.

Element interface. Each module implements a different interface.

Element behavior. The elements in this diagram represent the entire range of system behavior.

4.1.3 Context Diagram

This diagram shows the complete system.

4.1.4 Variation Guide

No variation shown in this view.

4.1.5 Architecture Background

This diagram was produced to help managers understand their dependencies on suppliers.

4.1.6 Other Information

No other information applies.

4.1.7 Related View Packets

No packets apply.

5 Component-and-Connector View

In this section, allocation and module views (Sections 2 and) are supplemented with an operational view (Figure 4) using a component-and-connector view type. This view provides a look at the computation flow through the system.

5.1 Component-and-Connector View Packet 1: Client/Server Interaction

The main activity in this system is the interaction between the client and the presentation layer

5.1.1 Primary Presentation

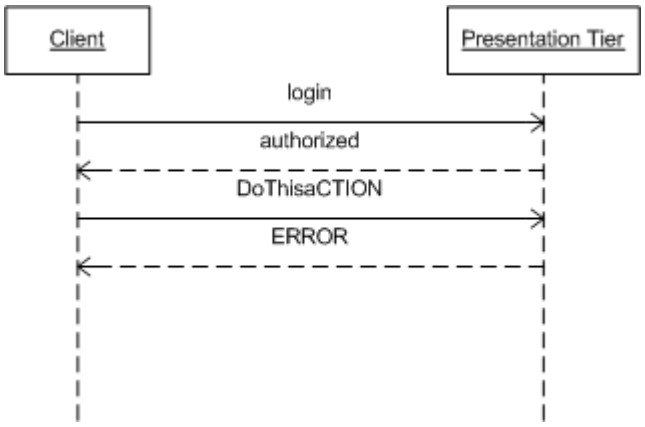

Figure 4: Interaction

5.1.2 Element Catalog

Elements and their properties. Table 4 displays element details.

Table 4: Elements and Responsibilities for Component-and-Connector View Packet 1: Client/Server Interaction

Element	Responsibilities
Client	**Handle user input and system output**
Presentation tier	**Accepts messages from client and either handles or forwards the message** **Formats the information to be sent to the user**

Relations and their properties. The relation shown in Figure 4 is method invocation.

Element interfaces.

- Social network interface
- Database interface
- Web server interface

Element behavior. No behavior applies.

5.1.3 Context Diagram

This interactions involve only top level modules shown in Figure 2.

5.1.4 Variation Guide

The variation in this scenario comes from the ability to replace implementations of interfaces.

5.1.5 Architecture Background

This interaction is part of the basic pattern expected in the 4 tier architecture design.

5.1.6 Other Information

No other information applies.

5.1.7 Related View Packets

No other packets apply.

6 Client Interface

6.1 Interface Identity

Client is shown in context in Figure 2

6.2 Resources Provided

```
system Client
      features
            receiveData: in event data port;
            requestData:out event data port;
            initiate:in event data port;
            completed:out event data port;
            error: out event data port;
      flows
            f1: flow path receiveData -> requestData{ latency => 50
            us;};
            f2: flow path initiate -> requestData;
            initiatePath:flow path initiate -> requestData{ latency =>
            10 us;};
            completedPath: flow path receiveData -> completed{ latency
            => 10 us;};

  end Client;
```

6.2.1 Resource Usage Restrictions

6.3 Locally Defined Data Types

Implementations of the Client interface will need to define process and thread types.

6.4 Error Handling

This version of Client provides only one visible error port. All errors are returned through this mechanism.

6.5 Variation Provided

Implementations can define many different variations on the basics provided they only use the ports visible on the interface. These variations are usually visible in the state machine that is implemented by the interface implementation.

6.6 Quality Attribute Characteristics

The Client is separated from the presentation layer for two reasons: (1) to enhance performance; (2) to enhance modifiability.

6.7 Element Requirements

The Client assumes there is a presentation tier that formats information in a particular manner so that the client can display it. This dependency must be coordinated between the person implementing the client and the person implementing the presentation tier.

6.8 Rationale and Design Issues

The Client was originally created to separate user and solution perspectives. Then the Client was decomposed into two pieces so that some of the client work could be done on the more powerful server machine.

6.9 Usage Guide

The Client is a typical client. It captures user interaction, interprets those based on the mode of the system and emits the appropriate event to the presentation layer. The state machine provided by the implementation will determine how a client is used.

.Here is one typical, minimal implementation:

```
system implementation Client.basic
      subcomponents
            clientProcess:process DefaultClientProcess;
            clientProcessor: processor platform::DefaultProcessor;
            clientMemory: memory platform::DefaultMemory;
            connections
                  conn1: event data port receiveData ->
                  clientProcess.put;
                  conn2: event data port clientProcess.get ->
                  requestData;
                  conn3: event data port initiate -> clientProcess.put;
                  conn4: event data port clientProcess.get ->
                  completed;
            flows
                  f1: flow path receiveData -> conn1 ->
clientProcess.thru -> conn2 -> requestData{ latency => 100 us;};
                  f2: flow path initiate -> conn3 -> clientProcess.thru
-> conn2 -> requestData { latency => 100 us;};
                  initiatePath:flow path initiate -> conn3 ->
clientProcess.thru -> conn2 -> requestData{ latency => 10 us;};
                  completedPath: flow path receiveData -> conn1 ->
clientProcess.thru -> conn4 -> completed{ latency => 10 us;};
            properties
                  Actual_Memory_Binding => reference clientMemory
                  applies to clientProcess;
                  Actual_Processor_Binding => reference clientProcessor
                  applies to clientProcess;

  end Client.basic;
```