

Presented by Scott Gardiner, Senior Consultant, Information Balance to the Ottawa Software Quality Association (www.osqa.org), Jan. 31, 2002, Ottawa, Canada.

Agenda

- Software Architectures
- The Zachman Architecture Framework
- Expanding Framework for Testing
- Using the Framework for Testing Artifacts
- Sample Project
- Q & A

Software Architectures

- At Information Balance, we use architectures to understand the key aspects of a system:
 - Information Architecture - data / structure, process / behaviour
 - Application Architecture - transforming data / process requirements into an application system
 - Technology Architecture - using specific technology products to implement the application systems

Software Architectures

- **We build architectures to:**
 - Better understand the software system being built
 - Identify opportunities for simplification and reuse
 - Communicate the structure and behaviour of the software system
 - Manage risk of building the system
- **We use an Architecture Framework to:**
 - Organize architectural components or artifacts
 - Ensure consistency of deliverables from project-to-project
 - Consistency assists in estimating project efforts

Zachman Framework (www.zifa.com)



ENTERPRISE ARCHITECTURE - A FRAMEWORK™

	DATA <i>What</i>	FUNCTION <i>How</i>	NETWORK <i>Where</i>	PEOPLE <i>Who</i>	TIME <i>When</i>	MOTIVATION <i>Why</i>	
SCOPE (CONTEXTUAL)	List of Things Important to the Business 	List of Processes the Business Performs 	List of Locations in which the Business Operates 	List of Organizations Important to the Business 	List of Events Significant to the Business 	List of Business Goals/Strat. Objectives/Success Factors 	SCOPE (CONTEXTUAL)
<i>Planner</i>	ENTITY = Class of Business Thing	Function = Class of Business Process	Node = Major Business Location	People = Major Organizations	Time = Major Business Event	Ends/Mean = Major Bus. Goal/Strat. Success Factor	<i>Planner</i>
ENTERPRISE MODEL (CONCEPTUAL)	e.g. Semantic Model 	e.g. Business Process Model 	e.g. Logistics Network 	e.g. Work Flow Model 	e.g. Master Schedule 	e.g. Business Plan 	ENTERPRISE MODEL (CONCEPTUAL)
<i>Owner</i>	Ent = Business Entity Rel = Business Relationship	Proc = Business Process IO = Business Resources	Node = Business Location Link = Business Linkage	People = Organization Unit Work = Work Product	Time = Business Event Cycle = Business Cycle	End = Business Objective Means = Business Strategy	<i>Owner</i>
SYSTEM MODEL (LOGICAL)	e.g. Logical Data Model 	e.g. "Application Architecture" 	e.g. "Distributed System Architecture" 	e.g. Human Interface Architecture 	e.g. Processing Structure 	e.g. Business Rule Model 	SYSTEM MODEL (LOGICAL)
<i>Designer</i>	Ent = Data Entity Rel = Data Relationship	Proc = Application Function IO = User Views	Node = IS Function (Processor, Storage, etc.) Link = Data Characteristics	People = Role Work = Deliverable	Time = System Event Cycle = Processing Cycle	End = Structural Assertion Means = Action Assertion	<i>Designer</i>
TECHNOLOGY MODEL (PHYSICAL)	e.g. Physical Data Model 	e.g. "System Design" 	e.g. "System Architecture" 	e.g. Presentation Architecture 	e.g. Control Structure 	e.g. Rule Design 	TECHNOLOGY CONSTRAINED MODEL (PHYSICAL)
<i>Builder</i>	Ent = Segment/Table/etc. Rel = Pointer/Key/etc.	Proc = Computer Function IO = Screen/Device Formats	Node = Hardware/System Software Link = Line Specifications	People = User Work = Screen Format	Time = Execute Cycle = Component Cycle	End = Condition Means = Action	<i>Builder</i>
DETAILED REPRESENTATIONS (OUT-OF-CONTEXT)	e.g. Data Definition 	e.g. "Program" 	e.g. "Network Architecture" 	e.g. Security Architecture 	e.g. Timing Definition 	e.g. Rule Specification 	DETAILED REPRESENTATIONS (OUT-OF-CONTEXT)
<i>Sub-Contractor</i>	Ent = Field Rel = Address	Proc = Language Stmt IO = Control Block	Node = Addresses Link = Pinpoints	People = Entity Work = Job	Time = Interrupt Cycle = Machine Cycle	End = Sub-condition Means = Step	<i>Sub-Contractor</i>
FUNCTIONING ENTERPRISE	e.g. DATA	e.g. FUNCTION	e.g. NETWORK	e.g. ORGANIZATION	e.g. SCHEDULE	e.g. STRATEGY	FUNCTIONING ENTERPRISE

Zachman Institute for Framework Advancement - (810) 231-0531
January, 2002

Copyright - John A. Zachman, Zachman International
Copyright Information Balance Inc.

This diagram shows the standard presentation of the Zachman framework.

The framework **partitions** the knowledge that defines an application system.

The **columns** of the framework provide the common sense rules that describe a story or explain an object. These rules are: Who? What? When? Where? How? and Why?

The **rows** of the framework provide increasing greater level of detail as required by various observers or listeners to a story. Some observers require only a very high level of detail, while some require an in-depth, technical description of the story.

The Zachman Framework is not the only framework available, but any framework that is adopted should be easy to explain to management and to IT development and testing professionals.

For more information on the Zachman Framework, refer to **www.zifa.com**.

Modified Framework

Architectural Level	Local View			Global View
	"What"	"How"	"Who"	"Where"
Scope (Contextual)	Statement of Work / Terms of Reference			
Information Architecture (Conceptual)	Conceptual Data Model	Process model	Interfaces - user and system	Locations
System/ Application Architecture (Logical)	Logical Data Model	Application Functions	System Roles, Accesses and Interface Requirements	Application Distribution
Technology Architecture (Physical)	Physical Database Architecture	Components and Platforms	Security Architecture / Integration Architecture	Network Architecture / System Distribution
Component View	Database DDL	Source Code	Security Definition / Interface Definition	Network Definition
Functioning Enterprise	Databases	System Components	System Security / System Interfaces	Network

Framework for understanding:
Development artifacts and transitions / traceability

January, 2002

Copyright Information Balance Inc.

6

This slide shows a modified framework showing a subset of the columns, from one of our projects. To make the slide more readable, we have left off the "When" and "Why" columns.

Columns:

The **What? column** describes the **Data** used in the application.

The **How? Column** describes the **Functions or processes** performed by the application.

The **Who? Column** describes the **people, roles and organizations** within the business processes supported by the application.

The **Where? Column** describes the **locations** in which the business processes are performed, on a global basis.

Rows:

The **Scope Row** presents the **high level vision** for the project. This is often contained in a Statement of Work or Terms of Reference document.

The **Conceptual Row** presents the **subject matter expert's understanding of the subject area**. This row contains the first modeling efforts.

The **Logical Row** presents **more detailed, logical models**. This row is still independent of the chosen technology.

The **Physical Row** extends the logical models to account for **targeted technology platforms**. This row prepares the model for implementation.

The **Detailed Representations Row** presents **source code**.

The **Functioning Enterprise Row** presents **executing systems and executable code**.

The last 2 rows may be combined into "Components".

What about QA / QC tasks ?

Can the Framework be used as a means of:

- Understanding QA / QC processes related to development artifacts?
- Understanding the completeness of QA / QC processes?

Add in Testing

	Local View			Global View	Static Test / Traceability	Dynamic Test / Traceability
Architectural Level	"What"	"How"	"Who"	"Where"		
Scope (Contextual)	Statement of Work / Terms of Reference				Requirements Test	User Acceptance Test (UAT)
Information Architecture (Conceptual)	Conceptual Data Model	Process Model	Interfaces - user and system	Locations		
System/ Application Architecture (Logical)	Logical Data Model	Application Functions	System Roles, Accesses & Interface Requirements	Application Distribution	Design Test	System Integration Test
Technology Architecture (Physical)	Physical Database Architecture	Components and Platforms	Security Architecture / Integration Architecture	Network Architecture / System Distribution	Construction Test / Code Review	Unit Test
Detailed Representations	Database DDL	Source Code	Security Definition / Interface Definition	Network Definition		
Functioning Enterprise	Databases	System Components	Security System / System Interfaces	Network		

Although shown as "columns" there QA / QC processes related to every cell in the framework.

We considered how to add Testing to the Framework. In this view, we consider Testing as an evaluative activity that evaluates the development products created in the other columns.

The **Static Testing column** outlines the static analysis that should be performed to verify the development products in each row.

The **Dynamic Testing column** describes the tests that should be executed to validate the development products in each row.

In fact, these columns bear a strong resemblance to the standard "V" model used to describe testing activities.

This view provides an excellent checklist of the scope of testing to be completed.

Add in Risk

Architectural Level	Local View			Global View	Risk
	"What"	"How"	"Who"	"Where"	
Scope (Contextual)	Statement of Work / Terms of Reference				Project Level Risks
Information Architecture (Conceptual)	Conceptual Data Model	Process model	Interfaces - user and system	Locations	Requirements / Business Risks
System/ Application Architecture (Logical)	Logical Data Model	Application Functions	System Roles, Accesses & Interface Requirements	Application Distribution	Architecture / Design Risks
Technology Architecture (Physical)	Physical Database Architecture	Components and Platforms	Security Architecture / Integration Architecture	Network Architecture / System Distribution	Software / Hardware / Vendor Risks
Detailed Representations	Database DDL	Source Code	Security Definition / Interface Definition	Network Definition	Component Level Risk
Functioning Enterprise	Databases	System Components	System Security / System Interfaces	Network	

Risks associated with each level can be more clearly understood.
Are you assessing Risk at all these levels?

Risk can be added to the framework. Project Level Risks and Component-Level Risks are part of the spectrum of risks through the levels of the project architecture.

Others?

How about:

- Configuration management?
- Metrics?

Use the Framework to understand the completeness of your QA / QC process relative to Development deliverables

Are there other views that can be added on to the Framework that will provide additional insights into various QA / QC aspects?

You probably didn't think you would have homework coming home from this presentation, but I would like you to consider how other aspects such as Metrics, Configuration Management or Problem Management could be added.

What about Testing Architecture?

Can the Framework be used as a means of classifying, organizing and understanding Testing artifacts?

What we have discussed up to this point is primarily extending the Development framework to show Testing / QA / QC activities.

However, Testing produces many products or artifacts of its own. Can we use this framework to describe the levels and transitions of testing artifacts that comprise a Testing Architecture?

Testing Architecture Framework

Motivation

- Although Testing is an evaluative process, it does create / produce products
- QA / Test analysts tend to have a focus on Process not Product
- A concern that Test Tools focus attention on the “physical” level too early

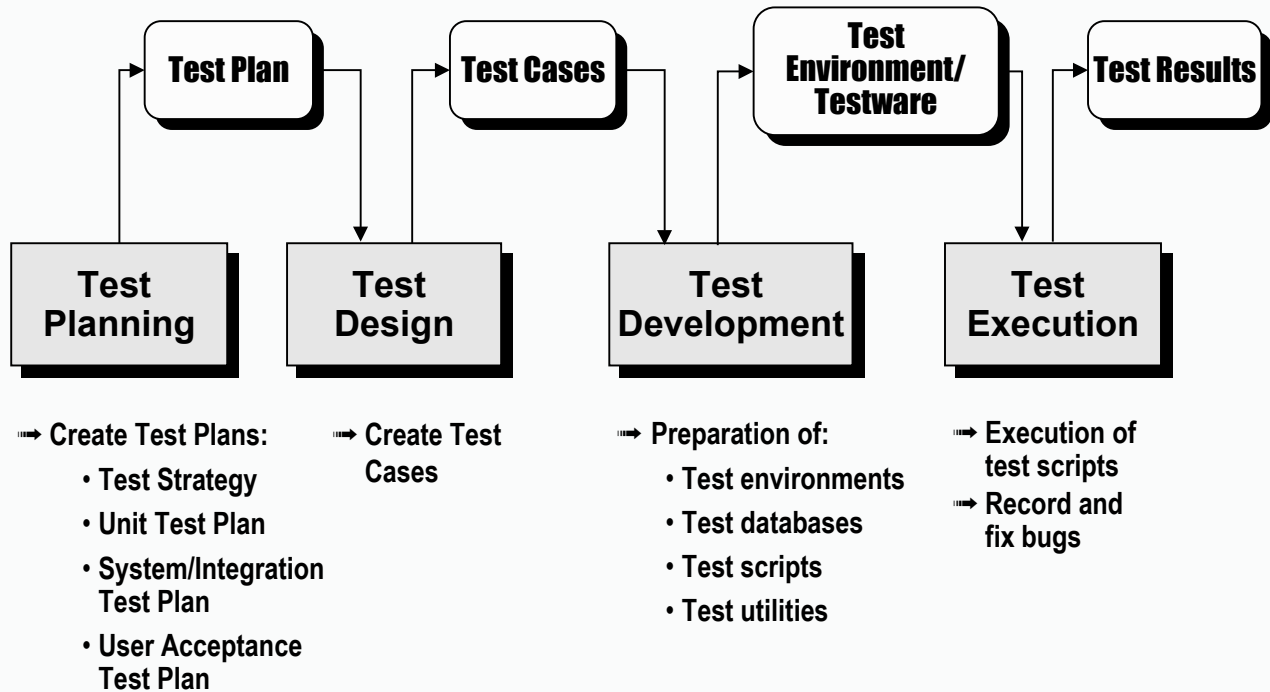
My original motivation for this presentation was a presentation given by Ed Kit at the EuroStar'99 conference in which he described a process of logically understanding test cases and then physically translating them into automated test cases.

My contribution then is to attempt to expand upon this concept to encompass all testing products.

Some other motivations for utilizing such a framework are:

- To understand “product” as well as “process”. QA / QC people are often completely focused on process, with little regard for products. However, Project Managers are keenly interested in the products to be produced by testing activities.
- As with all Tools, there is a tendency to rush into the “physical” level too early without first understanding what is needed.

Software Testing Process



January, 2002

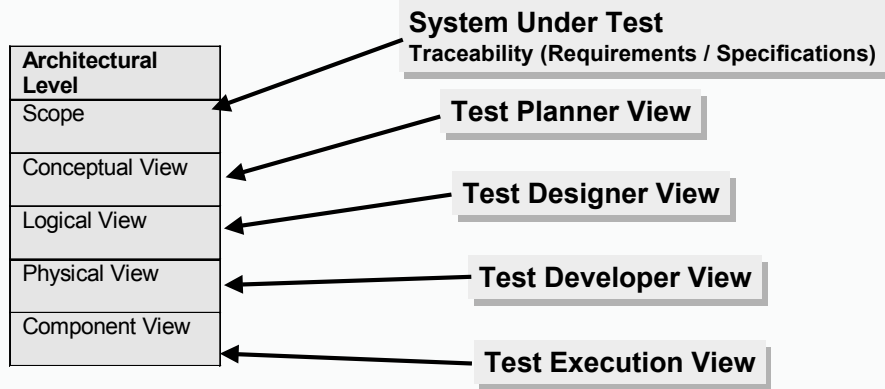
Copyright Information Balance Inc.

13

The Software Testing Process involves the planning, design, development and execution of all levels of software testing. The slide above depicts these major activities and the key products produced by these activities.

- Test Planning activities include the creation of test plans for each level of testing: the Master Test Plan, the Unit Test Plan, the System/Integration Test Plan and the User Acceptance Test Plan. These documents are critical to the success of the software development project as they serve the purpose of receiving clarification and consensus on “what” is to be tested.
- Test Design activities involve the creation of test cases. The test cases are the key to good testing.
- Test Development activities include the preparation of the test environments, the test databases and the test scripts.
- Test Execution activities include the execution and evaluation of the test cases. The goal is to ensure that expected result equals the actual result for all test cases.

Testing Architecture Levels



Testing Architecture Framework

Architectural Level	"What" (Data)	"How" (Process)	"Who" (Actors)	"Where" (Locations)	"When" (Events)	"Why" (Policies)
Scope	Requirements / Specifications (traceability target)					
Conceptual View (Test Planner)	List of Test Data Conditions	Test Models / Test Scenarios	List of System Actors, Roles and Interfaces	List of System Locations and Performance Requirements	List of Schedule Requirements	List of Test Standards & Measures
Logical View (Test Designer)	Test Data Definition	Functional Test Cases	Security and Interface Test Cases	Location / Test Environment definitions	Test Cycle definition	Test Design Reports
Physical View (Test Developer)	Test Database	Manual & Automated Test Scripts	Access authorities /System Interfaces for testing	Test Labs / Beta Sites, Performance Test Scripts	Test Event / Scheduler details	Test Development Reports
Functioning Tests (Tester)	Test Databases	Functional / Regression Test Components	Security, System Interface Test Components	Test Labs, Performance Test Components	Test Schedules and Events	Test Execution Results

This chart shows the aspects of a Testing Architecture Framework

January, 2002

Copyright Information Balance Inc.

15

This chart shows the overall view of the testing architecture and testing artifacts.

The **How? Column** describes the **Tests** to be performed by the application.

The **What? column** describes the **Data** used by the tests.

The **Who? Column** describes the **people, roles and organizations** who will be performing the tests.

The **Where? Column** describes the **locations** in which the testing will be performed.

The **When? (Testing) Column** describes the **schedules** for the testing activities.

The **When? (System) Column** describes the **schedule- or event-based testing** to be performed.

The following slides will explain these contents in detail.

Testing Architecture Framework Test Planner View

Information
Balance Inc.

Specific resource
planning aspects
for Who, When,
Where+

Architectural Level	"What" (Data)	"How" (Process)	"Who" (Actors)	"Where" (Locations)	"When" (Events)	"Why" (Policies)
Scope	Requirements / Specifications (traceability target)					
Conceptual View (Test Planner)	List of Test Data Conditions	Test Models / Test Scenarios	List of System Actors, Roles and Interfaces	List of System Locations and Performance Requirements	List of Schedule Requirements	List of Test Standards & Measures
Logical View (Test Designer)	Test Data Definition	Functional Test Cases	Security and Interface Test Cases	Location / Test Environment definitions	Test Cycle definition	Test Design Reports
Physical View (Test Developer)	Test Database	Manual & Automated Test Scripts	Access authorities / System Interfaces for testing	Test Labs / Beta Sites, Performance Test Scripts	Test Event / Scheduler details	Test Development Reports
Functioning Tests (Tester)	Test Databases	Functional / Regression Test Components	Security, System Interface Test Components	Test Labs, Performance Test Components	Test Schedules and Events	Test Execution Results

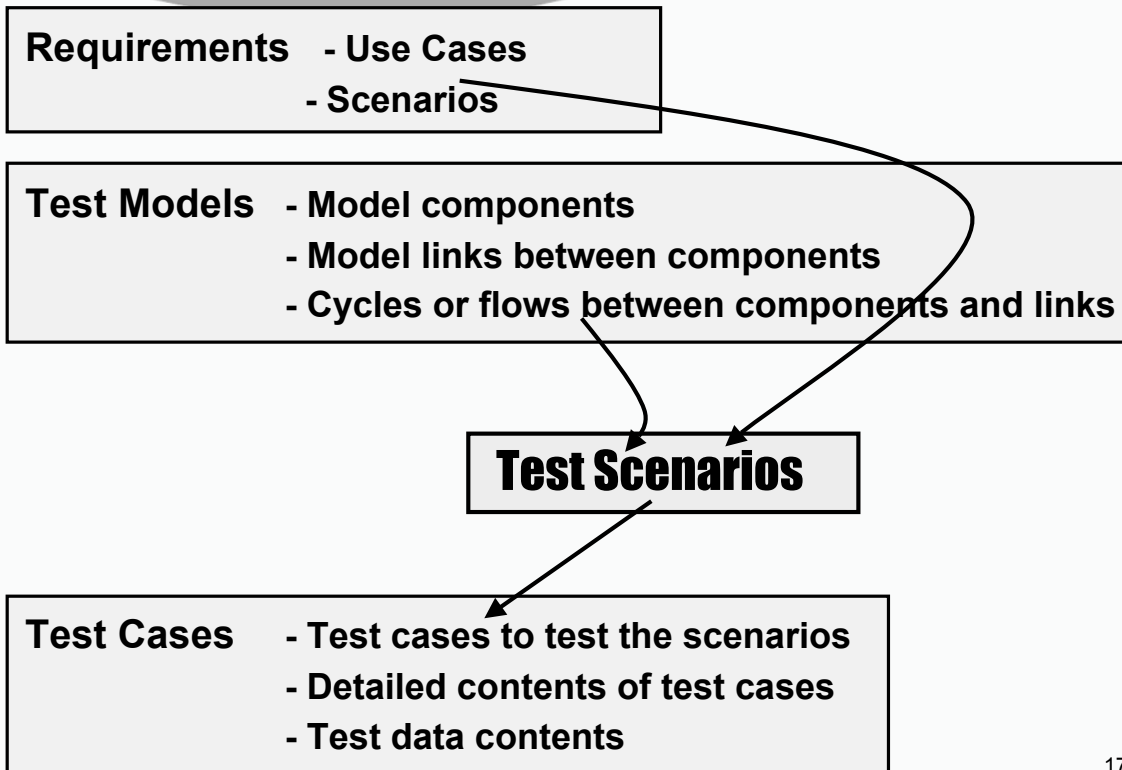
Conceptual View presents components of the Test Plan

January, 2002

Copyright Information Balance Inc.

16

Test Models & Techniques

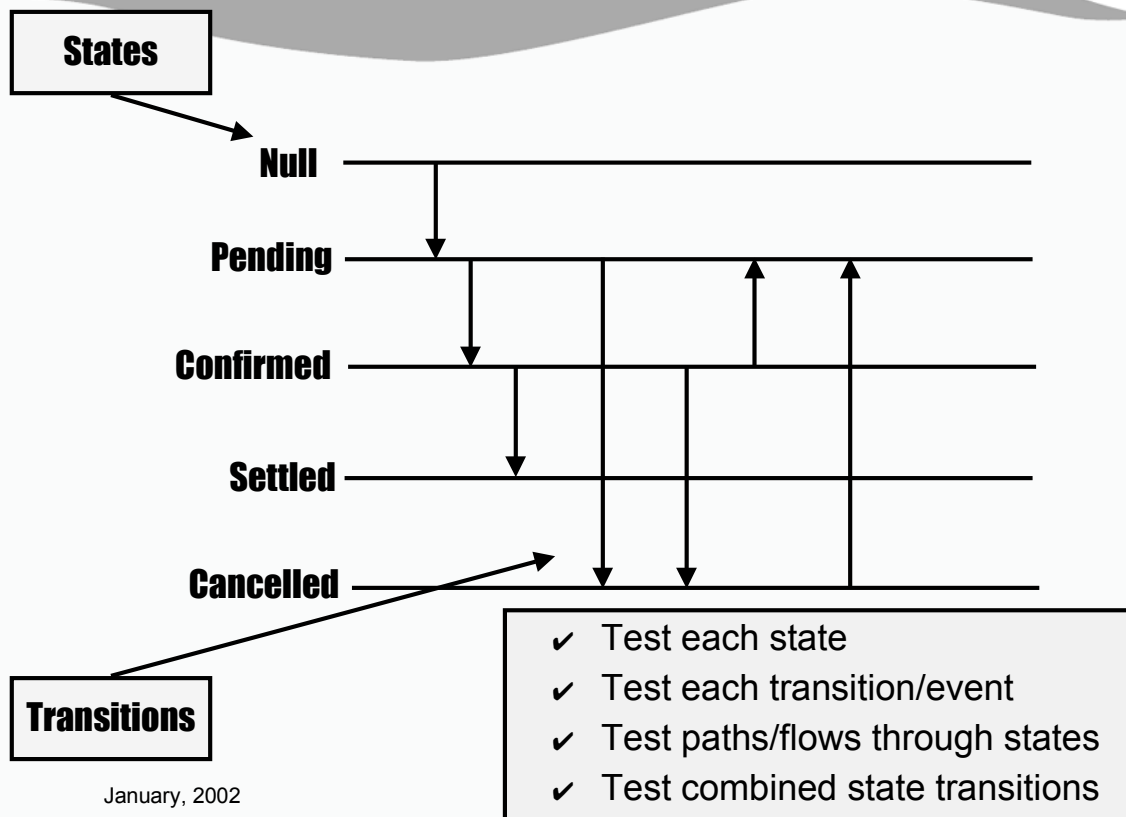


17

Key concepts on this slide:

- Requirements or specifications describe the behaviors required in the software system. These requirements must be unambiguous, consistent, complete, feasible, and testable. Test Cases are written based on these requirements and must be traceable directly back to these requirements.
- Use Cases or user scenarios present ways that users will actually use the system. This allows the software to be tested in realistic situations.
- Test Models represent the software system as a model of components and links between the components. User scenarios can then be used to select appropriate paths or flows through the components that represent user usage situations.

Test Models - State Transitions



Any component, object, or entity that has a state or status has a object life cycle. This life cycle moves through the various states or status' as external events or actions are executed on that object. The slide above shows the simplified example of a stock trade. The trade (object) starts in a *pending* state when it is first created, moves to a *confirmed* state when approved by one trader and then to a *settled* state when the originator of the trade also approves.

The diagram shows how the states change in response to the *events* (such as the approvals). There are also additional states such as *cancelled* that add additional paths through the states.

If a system or part of a system can be described through State Transition model, then test cases can be derived by testing:

- Each state.
- The transitions/events that cause the state to change.
- Various paths or flows through the various states.
- There can also be interrelated states (i.e. two objects which have related states. As an example, the transmission object in a car will only move from 'Park' to 'Drive' if the Engine object is 'on'.)

Remember to include invalid events/transitions in your test cases.

Other Examples of Test Models

- Full / partial Site Map for links / interfaces
- Workflow Model for role-to-role interaction
- Behaviour diagrams (sequence, activity) for class / component interactions

For more ideas, refer to
www.model-based-testing.org

Testing Architecture Framework

Test Designer View

Architectural Level	"What" (Data)	"How" (Process)	"Who" (Actors)	"Where" (Locations)	"When" (Events)	"Why" (Policies)
Scope	Requirements / Specifications (traceability target)					
Conceptual View (Test Planner)	List of Test Data Conditions	Test Models / Test Scenarios	List of System Actors, Roles and Interfaces	List of System Locations and Performance Requirements	List of Schedule Requirements	List of Test Standards & Measures
Logical View (Test Designer)	Test Data Definition	Functional Test Cases	Security and Interface Test Cases	Location / Test Environment definitions	Test Cycle definition	Test Design Reports
Physical View (Test Developer)	Test Database	Manual & Automated Test Scripts	Access authorities /System Interfaces for testing	Test Labs / Beta Sites, Performance Test Scripts	Test Event / Scheduler details	Test Development Reports
Functioning Tests (Tester)	Test Databases	Functional / Regression Test Components	Security, System Interface Test Components	Test Labs, Performance Test Components	Test Schedules and Events	Test Execution Results

Logical View provides a logical definition of Test Cases prior to creating Automated Test Scripts

Testing Architecture Framework

Test Developer View

Architectural Level	"What" (Data)	"How" (Process)	"Who" (Actors)	"Where" (Locations)	"When" (Events)	"Why" (Policies)
Scope	Requirements / Specifications (traceability target)					
Conceptual View (Test Planner)	List of Test Data Conditions	Test Models / Test Scenarios	List of System Actors, Roles and Interfaces	List of System Locations and Performance Requirements	List of Schedule Requirements	List of Test Standards & Measures
Logical View (Test Designer)	Test Data Definition	Functional Test Cases	Security and Interface Test Cases	Location / Test Environment definitions	Test Cycle definition	Test Design Reports
Physical View (Test Developer)	Test Database	Manual & Automated Test Scripts	Access authorities /System Interfaces for testing	Test Labs / Beta Sites, Performance Test Scripts	Test Event / Scheduler details	Test Development Reports
Functioning Tests (Tester)	Test Databases	Functional / Regression Test Components	Security, System Interface Test Components	Test Labs, Performance Test Components	Test Schedules and Events	Test Execution Results

Physical View presents Test Environments and introduces Automated Test Languages

Testing Architecture Framework

Test Execution View

Architectural Level	"What" (Data)	"How" (Process)	"Who" (Actors)	"Where" (Locations)	"When" (Events)	"Why" (Policies)
Scope	Requirements / Specifications (traceability target)					
Conceptual View (Test Planner)	List of Test Data Conditions	Test Models / Test Scenarios	List of System Actors, Roles and Interfaces	List of System Locations and Performance Requirements	List of Schedule Requirements	List of Test Standards & Measures
Logical View (Test Designer)	Test Data Definition	Functional Test Cases	Security and Interface Test Cases	Location / Test Environment definitions	Test Cycle definition	Test Design Reports
Physical View (Test Developer)	Test Database	Manual & Automated Test Scripts	Access authorities /System Interfaces for testing	Test Labs / Beta Sites, Performance Test Scripts	Test Event / Scheduler details	Test Development Reports
Functioning Tests (Tester)	Test Databases	Functional / Regression Test Components	Security, System Interface Test Components	Test Labs, Performance Test Components	Test Schedules and Events	Test Execution Results

Functioning Tests View presents the execution of functioning tests and the tracking / correction of bugs

Testing Architecture Framework

Sample Test Project

Architectural Level	"What" (Data)	"How" (Process)	"Who" (Actors)	"Where" (Locations)	"When" (Events)	"Why" (Policies)
Scope	Requirements / Use Cases (traceability target for UAT); User Interface Design / High Level Design (traceability target for SIT); Component Design / Detailed Design(traceability target for Unit Test).					
Conceptual View (Test Planner)	List of Test Data Conditions	Test Models / Test Scenarios	List of System Actors, Roles and Interfaces	List of System Locations and Performance Requirements	List of Schedule Requirements	List of Test Standards & Measures
Logical View (Test Designer)	Test Data Definition (Word)	Functional Test Cases (Excel)	Security and Interface Test Cases (Excel)	Test Environment definition / Performance Requirements (Word)	Test Cycle definition (Word)	Traceability Matrix (Excel), Design Scorecard (Excel)
Physical View (Test Developer)	Test Database (Oracle utilities)	Manual & Automated Test Scripts (JUnit, WinRunner)	Access authorities /System Interfaces for testing (JUnit, WinRunner)	Test Labs / Beta Sites, Performance Test Scripts (LoadRunner)	Test Event / Scheduler details (cron)	Development Scorecard (Excel)
Functioning Tests (Tester)	Test Databases (Oracle; DBDiff; Access for test verification)	Functional / Regression Test Components (WinRunner)	Security, System Interface Test Components (JUnit, WinRunner)	Test Labs, Performance Test Components (LoadRunner, Performance Monitors)	Test Schedules and Events (cron)	Problem Tracking (TestTrack), Execution Scorecard (Excel)

January, 2002

Copyright Information Balance Inc.

23

How many Frameworks?

- Do we need a Framework for each level of testing?
 - Unit, Integration, System, Acceptance
- Scope of each level of testing is different
 - As seen in Scope row of previous slide
- Test Models of each level are different
 - State Transition, Site Map, Workflow Model for Acceptance Test
 - Sequence / Activity diagrams for Unit Test
- Determine if a separate Framework provides more understanding

Use a framework to:

- Define and easily understand testing products
- To view testing products independent of tools and processes used to create them
- Understand transitions between products
- View testing as a manufacturing process requiring reuse of products

Should you use such a framework at your shop?

Many developers like to use the framework as a way of thinking about a problem. They don't necessarily use it for creating every development product, because they feel it is too confining.

I believe it is a good tool for thinking about your testing process, your testing products and the transitions between products. It is also a good framework for determining where reuse in the form of templates, checklists and reusable test cases would assist in making your testing practice more effective.

Questions?

January, 2002

Copyright Information Balance Inc.

26

Bibliography

- Zachman, J.A., *A Framework for Information Systems Architecture*, IBM Systems Journal, vol. 26, no. 3, 1987.
- Sowa, J.F. & J.A. Zachman, *Extending and Formalizing the Framework for Information Systems Architecture*, IBM Systems Journal, vol. 31, no. 3, 1992.
- Sharp, Dr. John K., *Zachman Framework*, The Journal of Conceptual Modeling, February 1999.
- Kit, Ed, *The New Frontier...Third Generation Software Testing Automation*, EuroStar'99 Conference, 1999.