Design Driven Testing for ArcGIS Server Development
A Java/Flex Example

Doug Rosenberg, ICONIX

Wolfgang Hall, Prakash Darbhamulla, Jim McKinney, Esri
What’s Design Driven Testing?

A rigorous and systematic approach to designing and testing software

Described in a book by Matt Stephens and Doug Rosenberg

This talk describes the book’s example project, a hotel mapping system, nicknamed “mapplet”, built with ArcGIS Server and deployed commercially on VRResorts.com
Design Driven Testing is *not* Test Driven Design

The premise of Design Driven Testing (DDT) is that *design should drive testing* as opposed to tests driving design (TDD)

TDD focuses heavily on unit testing

DDT includes automated support for generating unit tests from designs, but also supports acceptance testing at the scenario and requirement level

Unit testing catches “errors of commission” but not “errors of omission”, so both levels are needed
Errors of *commission*: whoops, I coded that wrong

Errors of *omission*: whoops, I forgot about that

With unit testing you can only test the stuff you thought about.

But those are the easy bugs to find and fix…it’s the stuff you *didn’t* think about that causes most of the trouble.
More about errors of Omission

*Whoops, I forgot about that requirement*

*Whoops, I forgot about that rainy day scenario*  
(exception, error, or infrequently used access path)

These errors tend to be the real troublemakers on your project

Unit testing does nothing to help find them
DDT addresses acceptance testing at two levels

We test against **business requirements**
We test against all sunny day/rainy day **scenario paths**
This testing is done by an independent QA group, not the developers
DDT also automates generation of unit tests (JUnit, etc.)

Unit testing is still very important
DDT generates JUnit, FlexUnit test code automatically
Test generation is done from both detailed and conceptual design
These tests are managed and run by developers
We used a real world project to demonstrate DDT

We expected massive skepticism from the TDD/"agile" community

We wanted to prove our ideas on a real, commercially viable software project

We had worked with Esri before on mapplet 1 (example project for “Agile Development with ICONIX Process”)

Esri stepped up to the plate again - the result: mapplet 2.0

mapplet 2 applies GIS technology to the travel industry

We invite you to compare the original use cases and requirements to the finished product
We designed mapplet using ICONIX Process

Define requirements
Model the problem domain
Storyboard the screens
Write the use cases
Drive the domain model into a class model following the use cases
Conceptual design on robustness diagrams
Detailed design on sequence diagrams
Unit test against conceptual and detailed designs
Acceptance test against use cases and requirements

It worked (just like it always does). Here’s the proof.
Mapplet requirements at a glance

- Worldwide maps
- Satellite imagery
- Flex UI
- Server side Java
- Quick/Advanced Search
- Easy to use
- Photography overlay
- Real time rate checking
- Fast performance
- Scalable
- Commercially deployed

In short, a real good test case for DDT
I can’t get no architecture, I don’t need no infrastructure
This just in: we can’t test against requirements unless they’re written down

(We actually organized them, prioritized them, and negotiated scope using them too, but don’t tell anybody.)
We drove the design from the problem domain, not from unit tests.
We couldn’t think of a clever name for QuickSearchWidget
We weren’t very clever with Address or POI either
We “wrote the user manual before writing the code”
First you storyboard the screens
Here’s the complete narrative text for the *Quick Search* use case:

**BASIC COURSE:**
The system displays the Quick Search Widget, which contains several options for specifying an Area of Interest (AOI).
- User double clicks on a Bookmark: Invoke Use Bookmark
- User clicks Locate button on Address panel: Invoke Use Address
- User clicks Locate button on POI Panel: Invoke Use Point of Interest
- User clicks push-pin icon on Location Panel: Invoke Use Map Location Picker
The system updates the map extent and AOI based on the location. AOI is smaller than “Local View” limit.
- System searches the static geodatabase for hotels within the specified AOI.
- Invoke Display Hotels on Map
- Invoke Display Hotels in List View

**ALTERNATE COURSE:**
- **AOI greater than “Local View” limit:** System displays message: “Please zoom in further to display hotels.”
- **No Hotels Found (and AOI within Local View limit):** System displays message: “No hotels found.”
- **“Show Landmarks” button is enabled:** System searches the geodatabase for Landmarks within the specified AOI and displays them on the map.
- **User selects Pan/Zoom tools:** Invoke Zoom-Pan on Map
Quick Search finds all hotels in Waikiki - quickly
Advanced Search lets you zero in on what you want.
BASIC COURSE:
The system enables the Advanced Search widget when an AOI exists of “local” size. The
user clicks the Advanced Search icon; the system expands the Advanced Search widget
and populates Check-in/Check-out fields with defaults. The user specifies the the
Reservation Detail including Check-in and Check-out dates; the system checks that
the dates are valid. The user selects the number of adults from the drop-down menu
(range: 1–6), and then enters the number of rooms from the drop-down menu (range:
1-number of adults). Optionally, the user selects desired amenities, price range, star
ing, hotel chain, and the “hot rates only” check box. The user clicks FIND. The
system searches for hotels within the current AOI, producing a Hotel Collection.
Then invoke Display Hotels on Map and Display Hotels on List Widget.

ALTERNATE COURSES:
Check-out date prior to Check-in date: The system displays a user-error dialog:
“Check-out date prior to Check-in date.”
Check-in date prior to today: The system displays a user-error dialog: “Check-in date
is in the past.”
System did not return any matching hotels: The system displays a message: “No
hotels found.”
Conceptual design tells us the “logical functions” that need to be tested
We generate a test case for each “controller”
Then we specify the test details
This test caught a bug before mapplet was released

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Description</th>
<th>Input</th>
<th>Acceptance Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check-in date earlier than today</td>
<td>Validation should fail on the dates passed in.</td>
<td>Check-in date: Yesterday</td>
<td>The dates are rejected as invalid.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Check-out date: Any</td>
<td></td>
</tr>
<tr>
<td>Check-out date earlier than check-in date</td>
<td>Validation should fail on the dates passed in.</td>
<td>Check-in date: Tomorrow</td>
<td>The dates are rejected as invalid.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Check-out date: Today</td>
<td></td>
</tr>
<tr>
<td>Check-out date same as check-in date</td>
<td>Validation should fail on the dates passed in.</td>
<td>Check-in date: Today</td>
<td>The dates are rejected as invalid.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Check-out date: Today</td>
<td></td>
</tr>
<tr>
<td>Good dates</td>
<td>Validation should succeed for the dates passed in:</td>
<td>Check-in date: Today</td>
<td>The dates are accepted.</td>
</tr>
<tr>
<td></td>
<td>Check-out date is later than the check-in date, and the check-in date is later than yesterday</td>
<td>Check-out date: Tomorrow</td>
<td></td>
</tr>
</tbody>
</table>
Transforming the test case to FlexUnit
import flexunit.framework.*;

class DatesAreCorrectTest
{

    [Before]
    public function setUp(): void
    {
        // set up test fixtures here...
    }

    [After]
    public function tearDown(): void
    {
        // destroy test fixtures here...
    }

    /**
     * Validation should fail on the dates passed in.
     * Input: Check-in date: Yesterday
     *        Check-out date: Any
     * Acceptance Criteria: The dates are rejected as invalid
     */
    [Test]
    public function checkInDateEarlierThanToday(): void
    {
Can I do design driven testing without the design?

No.
Because we generate the test cases from the design.
Then we transform the test cases to test classes
And auto-generate the unit test code
Test criteria propagate into the JUnit code as comments

```java
@Before
public void setUp() throws Exception {
   // set up test fixtures here...
}

@After
public void tearDown() throws Exception {
   // destroy test fixtures here...
}

/**
 * Specify a 5 star hotel in NY City for $50.
 * Should result in an XML response containing zero hotels.
 *
 * Input: Hotel Filter with:
 *     starsMin: 5
 *     starsMax: 5
 *     location: New York
 *     priceMin: $50
 *     priceMax: $50
 *
 * Acceptance Criteria: Should survive parsing the zero-hotel
 * XML result and return an empty HotelCollection.
 */
@Test
public final void noHotels() throws Exception {
}

/**
 * Search for hotels in Waikiki that have swimming pools.
 * Should result in an XML response containing LOTS of matching hotels.
 *
 * Input: Hotel Filter with:
 *     Location: Waikiki
 *     Amenities: Swimming Pool
 *```
We need breadth and depth in unit test coverage

Testing against conceptual design gives breadth

Testing against detailed design gives depth

Sometimes testing against the conceptual design is enough

This is “testing smarter, not harder”
Developer testing catches most coding errors

Easy to figure out what to test

Generating tests from designs saves coding time

Testing against conceptual design allows us to get really good test coverage with fewer tests

But what about those errors of omission?
Acceptance testing: expanding “use case threads”

If we have one sunny day scenario and three rainy day scenarios, we need at least four “thread tests”

One thread test for the sunny day scenario all the way through

One thread test for each rainy day scenario including some steps from the sunny day path and the steps of the rainy day path

QA uses these thread tests to catch stuff the developers forgot

We can now generate the thread tests automagically
Start with a narrative use case

Search by address

The user types an address using all address fields on the Quick Search window. The system enables the "Locate" button as soon as an entry was made in either one of these fields: City, State, Postal, Country.

The user clicks "Locate". The system geocodes the location based on the level of detail provided by the user and stores any candidates in an Address Candidate Collection. If single candidate is found or exactly one of the multiple candidates has a 100% match rate, the System sets the AOC based on this Address Candidate.

ALTERNATE COURSES

User clicks "Clear": Entries in any fields will be cleared

Multiple valid candidates found: The System displays an Address Candidate widget with a list of potential candidates to chose from. The user selects an Address Candidate.

No candidates were found: the system displays a message "Location not found"
Transform to a “structured scenario”

And then magic happens...
Auto-generate an activity diagram

This lets us check that we’ve got the scenario specified correctly.

Pay particular attention to “Multiple Valid Candidates found”
Auto-generate thread tests

The user types an address using all address fields on the Quick Search window. The system enables the "Locate" button as soon as an entry was made in either one of these fields: City, State, Postal, Country.

The user clicks "Locate". The system geocodes the location based on the level of detail provided by the user and stores any candidates in an Address Candidate Collection. If single candidate is found or exactly one of the multiple candidates has a 100% match rate, the system sets the AOI based on this Address Candidate.

**ALTERNATE COURSES**

User clicks "Clear": Entries in any fields will be cleared.

Multiple valid candidates found: The system displays an Address Candidate widget with a list of potential candidates to chose from. The user selects an Address Candidate.

No candidates were found: The system displays a message "Location not found".

**Test scripts**

- Scenario: (Not Run) Basic Path
- Scenario: (Not Run) Multiple valid candidates found
- Scenario: (Not Run) No candidates were found
- Scenario: (Not Run) The user clicks "Clear"

Try doing this by hand sometime
Each thread test is auto-populated with steps
QA gets very useful test scripts, very quickly

Try doing this by hand on a system with 50 use cases

<table>
<thead>
<tr>
<th>Name</th>
<th>Object</th>
<th>Description</th>
</tr>
</thead>
</table>
| No candidates were found:      | Use Address_TestCase1 | Alternate: When [No candidates were found]:  
5b.1. The system displays a message “Location not found” on the Quick Search Window.  
Result: No candidates were found: complete.  
Use case ends.                     |
| Multiple valid candidates found: | Use Address_TestCase1 | Alternate: When [Multiple valid candidates found]:  
5a.1. The system displays an Address Candidate widget with a list of potential candidates to choose from.  
5a.2. The user selects an Address Candidate from the Address Candidate widget.  
Result: Multiple valid candidates found: complete.  
Rejoin at 5.                     |
| The user clicks “Clear”        | Use Address_TestCase1 | Alternate: When [The user clicks “Clear”]:  
1a.1. Entries in any fields on the Quick Search Window will be cleared. |
And here’s why developer testing isn’t enough

We use 2 different search services.

The address search service expected a complete address.

The use case said we enable the locate button as soon as text is entered in any address field.
Whoops! We expected multiple candidates, we got no results found.

What the user expects.

What we got.

We fixed this bug BEFORE the software got released.
Acceptance testing: testing against requirements

Auto-generate test cases from requirements
Unintended side-effects of de-cluttering

Default check-in/check-out dates caused an unintended side effect with advanced search

Too many hotels got removed from the map

Solution: use **mm/dd/yy** instead of default date

<table>
<thead>
<tr>
<th>Name</th>
<th>Requirement / Description</th>
<th>Input</th>
<th>Acceptance Criteria / Result Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test for no side-effects</td>
<td>31. hotel filter -- hotel query by Amenities</td>
<td>Area of Interest (at a “local” map scale)</td>
<td>All hotels on the decluttered map display must match the selected filter criteria.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Initial map display (showing hotels within AOI)</td>
<td>Verify that amenity selection does not remove too many hotels from the map (i.e., filtering on “Room Service” shouldn’t inadvertently affect other amenities).</td>
</tr>
</tbody>
</table>

| Filtering by sets of amenities | 31. hotel filter -- hotel query by Amenities | Area of Interest (at a “local” map scale) | All hotels on the decluttered map display must match the selected filter criteria. |
| | A series of “visual inspection” tests will be conducted to verify that amenity filtering is working properly. Each of these tests will involve checking various sets of amenities (for example “Pool, Room Service, | Initial map display (showing hotels within AOI) | Deducted map display (showing a filtered subset of Hotels on the map). |
Both scenario testing and requirement testing caught bugs before release

Mapplet has been deployed for several months, and there hasn’t been a bug-fix release, patch, or service pack
Technical Challenge

Data volume restrictions:
- Hotel service provider has restrictions on number of dynamic hotel information requests
- Our approach was not the typical workflow where users select a particular geographic area and then request a small number of hotels. We wanted to map hotels at both rural and urban extents and still provide sufficient hotel information.
- We also wanted the user to pan freely to explore hotels graphically, thus resulting in a much higher number of records queried

Solution:
- Hybrid implementation with static data and dynamic data sources based on user workflow.
Solution Architecture

**Flex Client**

- Hotel Info Web Service (HTTP (XML))
- Hotel Info FTP
- ArcGIS Online Map Services (HTTP (REST/JSON))
- POI Service (HTTP (REST/JSON))
- Address Geocoding Service (HTTP (REST/JSON))

**Mapplet Server**

- Search Hotels Service
- ArcGIS Server
- Geodatabase (FGDB)

**Dynamic data**

- Download
- Static/stored data
UI Design Challenge

Challenge:
- Provide a user interface that does not require any expert knowledge (i.e.: my mother should be able to use it with no instructions)
- It is really hard for most developers to imagine a workflow for users with no application knowledge.

Solution:
- Perform holistic testing
- Bring 10 users who have no prior knowledge of the application into a test lab and ask them to perform simple scenarios (e.g. find a hotel in Palm Springs with 3-star rating under $150/night)
- Capture all problems experienced and treat them as real problems
A few details about the development cycle

• Esri’s Flex Viewer samples made it really easy to start prototyping

• We almost got carried away with prototyping and almost got ahead of the design; we had to put on the brakes

• Using the DDT process properly is time-consuming prior to application development but more than makes up for lost time during development and particularly during testing

• The comprehensive UML design allowed us to switch developers mid-project without any problems
Summing up

Driving design from testing is backwards

Driving design from domain models and use cases works reliably

Driving testing from design catches bugs

Try mapplet at http://hotelmaps.vresorts.com

Read the use cases in Design Driven Testing

Compare the use cases to the finished product.
Additional Resources

Web sites:
- http://www.iconixsw.com

Contact:
- doug@iconixsw.com
- whall@esri.com
- UMLTraining@iconixsw.com