An Approach to Agile Automation Testing

QAANOVA
September 16, 2008
Frank Hurley, Technical Manager
Agile development is here

- Iterative development and deployment
- Faster times to market/client stakeholders
- Proven success - ?

“77% of respondents indicated that 75% or more of their agile projects were successful” – Scott Ambler, The Agile Edge, Dr. Dobbs Journal, August 2007

Automated testing is key to agile

One successful approach
About Cigital

- Security solutions
  - Assessments / Training / more
- QA solutions
  - Testing management / Custom test automation / more
Agenda

- Agile development
- QA’s role
- Unit testing vs. QA testing
- Automated QA testing
- An approach to automated QA testing in an agile environment
  - Framework
  - Domain-specific language for test definition
- Conclusions
Agile development

- Started in 1990’s
  - Extreme Programming (XP) – Kent Beck
  - Scrum – Ken Schwaber, among others
  - Agile Manifesto (2001)
    - Brian Marick (former Cigitalite!)
- Principles of agile include:
  - Frequent delivery/deployment
  - Customer collaboration
  - Progress measured by working software
- Unit testing
  - Safety net, allows refactoring “courage”
Before agile, there was Waterfall
- One development effort
- One “over the wall” delivery to QA team
- Spiral, other early iterative: repeat a few times
- Automation testing: an afterthought

With agile
- No more “over the wall”
- Development and testing done in parallel
  - …as currently done with unit testing
- Testing is automated from the start
  - No time consuming manual regression testing
Unit testing vs. QA testing

- Unit testing
  - Goal: code coverage
  - Automated
  - Typically done by development

- QA testing
  - Goal: requirements coverage
  - Real goal: ensure solid software that meets users’ needs
  - Other real goal: integration testing
  - Often manual, some automation
  - Typically done by “testers”
What is a “tester”?

- Testers and Developers Think Differently
  - Bret Pettichord, STQE Magazine, Jan/Feb 2000
  - Good Developers
    - Model system design
    - Knowledge of product internals
    - Focus on how it can work
  - Good Testers
    - Model user behavior
    - Domain knowledge
    - Focus on how it can go wrong

- Need best of both worlds for an automated QA solution
Need best of both worlds

- Case in point: Microsoft Vista
  - Replaced non-coding testers with SDET’s (Software Development Engineers in Test)
    
    Source: Joel Spolsky, Talk at Yale CS Dept., http://www.joelonsoftware.com/items/2007/12/03.html
  - Result: ?

- Bottom line: We need both developers and testers to build effective automated QA testing
  - Need developers (SDET’s) to code it
  - Need testers to somehow tell what to code
Automated QA testing

- Tools for GUI automation
  - Quick Test Pro, WinRunner
  - Others: Silk, Visual Test, QARun
- Phases
  - Record and Play
    - Looks so simple
    - Gets complicated quickly
  - Scripting
    - Requires low level details
    - Not scalable if not organized correctly
  - Data Driven
    - Getting better – testers can define data set
    - Not scenario driven
Automated QA testing

The Problem

- Automation testing requires development
- Development requires developers
  - Developers (i.e. “automation experts”) = big $$$
- Development requires decent development tools

“…testers had to jump through extra hoops to do basic things when they were saddled with crummy languages.” – Bret Pettichord, Homebrew Test Automation, September 2004

It would be nice if…

- Testers could write tests
- Developers could write the automation code
- Combine the two with little effort
An approach: the framework

- The framework
  - Testers write tests in high level domain-specific language
  - QA developers (aka “Automation experts”, SPET’s) write specific “test commands”
  - Integration → combines the two

- Domain-specific language (DSL)
  - Easy to learn
  - Business-specific
    - No techie required!
Domain-specific language (DSL)

```xml
<testSuite name="Reservation tests" >
  <test name="Book-Flight-001" testcaseid="FLT-01" >
    <login username="TestAdmin27" />
    <addCustomer firstName="John" lastName="Doe"
      address="123 Main Street"
      city="Manassas" state="VA" />

    <bookFlight fromAirport="IAD" toAirport="HNL" />

    <verifyFlights>
      <flight no="3423" depart="10:15" arrive="17:22" />
      <flight no="6342" depart="11:20" arrive="19:04" />
      <flight no="7311" depart="14:45" arrive="22:43" />
    </verifyFlights>

    <teardown><logout /></teardown>
  </test>

  ...
</testSuite>
```
DSL hides implementation details

```java
public class LoginUI extends DefaultUI {
    public void goTo(String menu) {
        selenium.selectFrame("relative=top");
        String menuId = menu.replaceAll(" ", "");
        String menuLocator = "//a[@href, '"' + menuId + '"'] and @class='SignPost']";
        selenium.click(menuLocator);
    }
}
```

```java
public class AddCustomerUI extends DefaultUI {
    public void openCornerMenu() {
        selenium.selectFrame("relative=top");
        String menuItem = "//div[@class='MenuItem']";
        selenium.mouseOver(menuItem);
        String subMenuItem = "//div[@class='SubItem']";
        selenium.waitForElementPresent(subMenuItem);
        selenium.click(subMenuItem);
        String findNeutralForm = "//div[@"]
        selenium.waitForElementPresent(findForm);
    }
}
```

```java
public class BookFlightUI extends DefaultUI {
    protected void execute(Context ctx) throws Exception {
        if (getName().contains("minimize")) { minimize(); }
        else if (getName().contains("maximize")) { maximize(); }
        else if (getName().contains("goto")) { goTo(); }
        else if (getName().contains("add")) { addProperty(); }
        else if (getName().contains("edit")) { editProperty(); }
    }
}
```

```xml
<testSuite name="Reservation">
    <test name="Book-Flight-001" testcaseid="FLT-01">
        <login username="TestAdmin27"/>
        <addCustomer firstName="John" lastName="Doe" address="123 Main Street" city="Manassas" state="VA"/>
        <bookFlight fromAirport="IAD" toAirport="HNL" />
        <verifyFlights>
            <flight no="3423" depart="10:15" arrive="17:22" />
            <flight no="6342" depart="11:20" arrive="19:04" />
            <flight no="7311" depart="14:45" arrive="22:43" />
        </verifyFlights>
    </test>
</testSuite>
```
Old way

- Development team:
  - Create Iteration 1 application code
  - Create Iteration 2 application code
  - Create Iteration 3 application code
  - Fix I1 bugs
  - Fix I2 bugs

- QA team:
  - Test planning activity
  - Manual testing of Iteration 1 code
  - Manual testing of Iteration 2 code
  - Bugs found
  - Bugs found

- Automation team:
  - Automation testing develops independently of iteration plan

Timeline:
- Start of development
- End of Iteration 1
- End of Iteration 2
- End of Iteration 3
New way

Development team

Create Iteration 1 application code

Continuous Integration

Create Iteration 2 application code

Continuous Integration

Create Iteration 3 application code

Continuous Integration

QA Testers

Create test scripts for Iteration 1 code

Automation team

Create test commands for Iteration 1

Create test scripts for Iteration 2 code

Create test commands for Iteration 2

Create test scripts for Iteration 3 code

Create test commands for Iteration 3

Start of development

End of Iteration 1

End of Iteration 2

End of Iteration 3
Pros and Cons

- **Pros**
  - **Cost Savings**
    - 1 Automation Expert : 2-3 Test Scripters
    - Instead of 2-3 automation experts
  - **Time Savings**
    - Parallel Development
  - **Stable test scripts**
    - If GUI changes, test scripts stay the same
      - Test commands need to change
  - **Simple/maintainable test command code**
Pros and Cons

- **More Pros**
  - Automation tests from the start
  - Tests are authored by testers, not developers

- **Cons**
  - Tight synchronization between dev team and QA team can be a challenge
  - Normal GUI automation issues
    - Timing / data setup / etc.
    - Requires “pair debugging”
  - In reality: Some overlap in old way/new way
Summary

- Agile software development requires automated testing
- Automated QA testing requires skills from both tester and developer
- The trick is to use the skills from both groups in an efficient and effective manner

- This framework provides an avenue to allow:
  - Testers to concentrate on test scenarios without worrying about low-level details
  - QA developers to implement test commands without worrying about business domain
  - **Big time/cost savings**
Where to Get More Information

- Agile Automated Test Framework
  - Frank Hurley <frank.hurley@cigital.com>
  - Drew Kilbourne <drew.kilbourne@cigital.com>
  - Stuart Dross <stuart.dross@cigital.com>
  - Terri Randolph <terri.randolph@cigital.com>