Test Driving User Interfaces

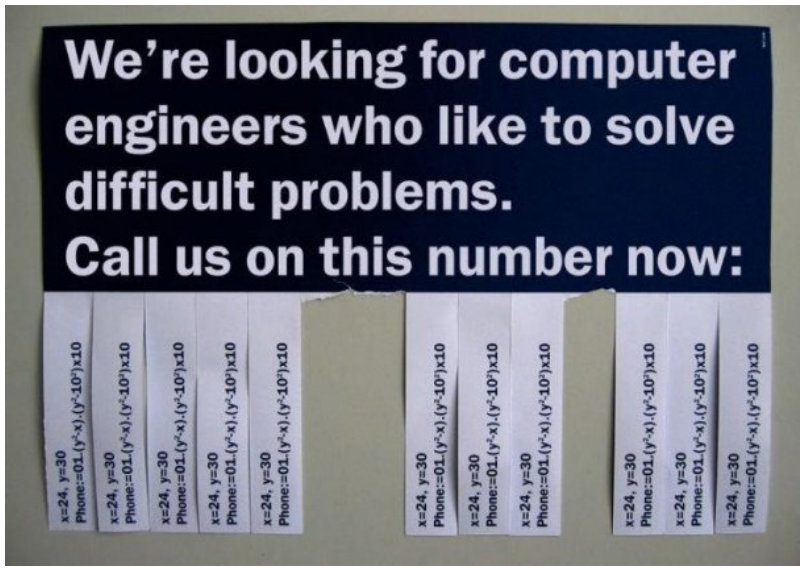
Anthony Williams

Just Software Solutions Ltd
http://www.justsoftwaresolutions.co.uk

3rd September 2015

Test Driving User Interfaces

User Interfaces are widely considered to be one of the hardest aspects of software to develop with TDD.



Test Driving User Interfaces

If it hurts, do it more often!

Test Driving User Interfaces

- What does it mean to Test-Drive a UI?

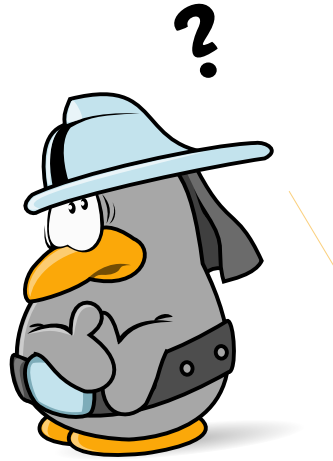
Test Driving User Interfaces

- What does it mean to Test-Drive a UI?
- Why is it considered to be hard?

Test Driving User Interfaces

- What does it mean to Test-Drive a UI?
- Why is it considered to be hard?
- How can we do it?

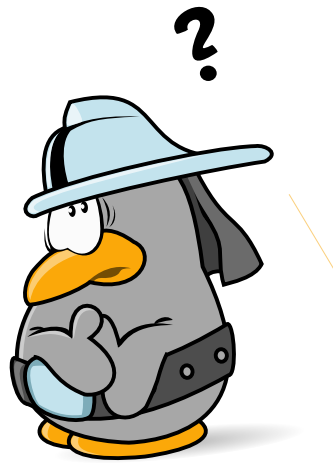
What does it mean to Test-Drive a User Interface?



What does it mean to Test-Drive a User Interface?

1. Write a test for the user interface
2. Add the necessary code and UI features to make it pass
3. Refactor
4. Go to step 1

Why is Test-Driving User Interfaces considered to be hard?



Why is Test-Driving User Interfaces considered to be hard?

- People don't know how to write UI tests

Why is Test-Driving User Interfaces considered to be hard?

- People don't know how to write UI tests
- Frameworks that don't support testing

Why is Test-Driving User Interfaces considered to be hard?

- People don't know how to write UI tests
- Frameworks that don't support testing
- Code that doesn't support testing

Why is Test-Driving User Interfaces considered to be hard?

- People don't know how to write UI tests
- Frameworks that don't support testing
- Code that doesn't support testing
- Tests written after the code

Why is Test-Driving User Interfaces considered to be hard?

- People don't know how to write UI tests
- Frameworks that don't support testing
- Code that doesn't support testing
- Tests written after the code
- It's an unfamiliar way of working

Aside: Seams

Aside: Seams

Michael Feathers defines a seam as:

> *. . . a place where you can alter behaviour in your program without editing in that place.*

Aside: Seams

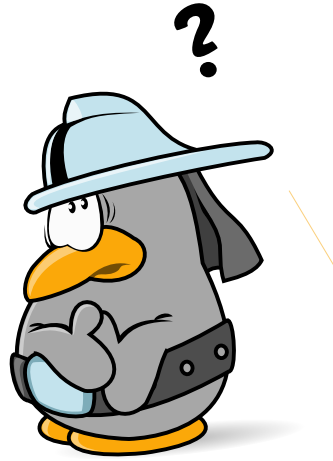
Seams allow us to separate the code under test from the rest of the system.

Aside: Seams

Typical Seams:

- Callbacks
- Virtual Functions
- Interfaces

How can we Test-Drive User Interfaces?



How can we Test-Drive User Interfaces?

TDD works best when the tests are in the same language as the code under test.

How can we Test-Drive User Interfaces?

TDD works best when the tests are in the same language as the code under test.

This is the easiest way to get a seam in the right place.

Framework Support

Test-Driving User Interfaces is easiest when the framework supports you.

Framework Support

Test-Driving User Interfaces is easiest when the framework supports you.
JavaScript frameworks often allow you to trigger events in code.

Framework Support

Test-Driving User Interfaces is easiest when the framework supports you.
JavaScript frameworks often allow you to trigger events in code.
Desktop frameworks often don't.

Examples

Two examples:

- JavaScript/JQuery
- C++/MFC for Windows

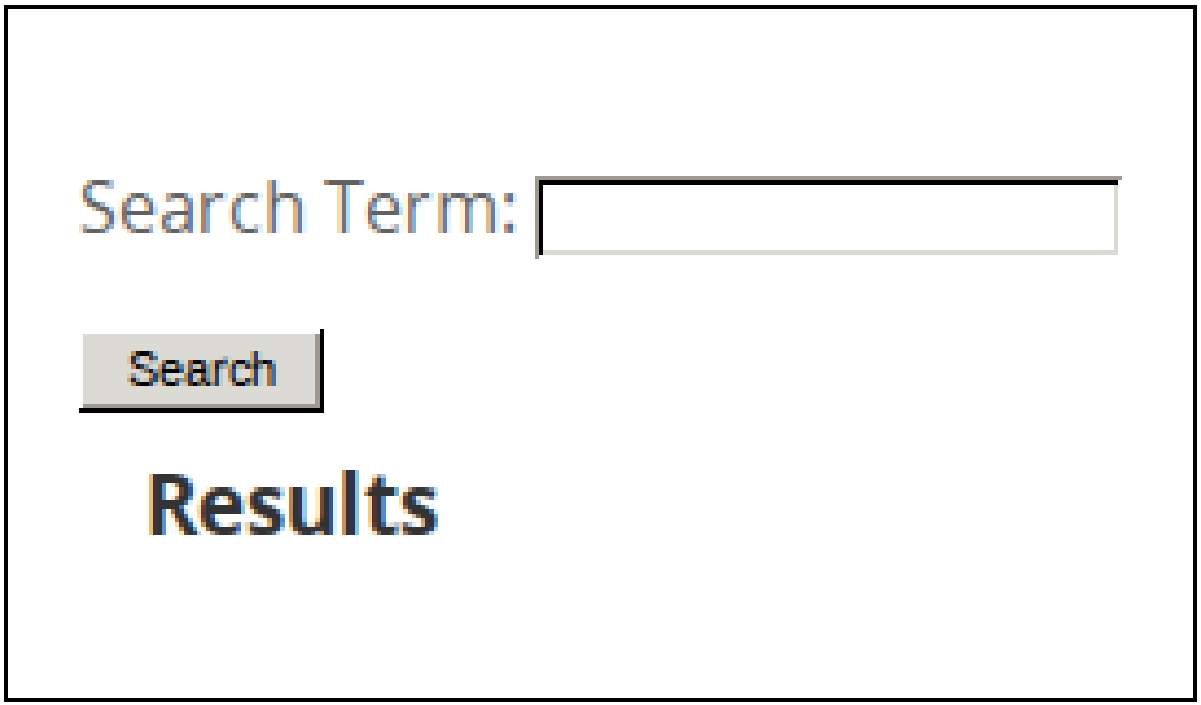
JavaScript Example

JavaScript Example

Suppose we have a web-app with an AJAX-based search box.

The user enters a search term and clicks “search”, and the app does a background search and displays the results.

JavaScript Example: Mockup



JavaScript Example: Important Points

1. Ignore styling for now: **we're not testing the CSS**.

JavaScript Example: Important Points

1. Ignore styling for now: **we're not testing the CSS**.
2. We need to create the HTML fragment in our test.

JavaScript Example: Important Points

1. Ignore styling for now: **we're not testing the CSS**.
2. We need to create the HTML fragment in our test.
3. We need to trap the AJAX call: we're testing the UI, not the backend.

JavaScript Example: First test

```
function test_edit_box_exists(){
  var body=$('body');
  body.empty();
  body.append(
    load_html_from_file('fragments/search_form.html'));
  var edit_box=body.find('form input.search-term');
  if(edit_box.length!=1){
    alert('There are '+edit_box.length+
      ' edit boxes, instead of 1');
    return false;
  }
  return true;
}
```

JavaScript Example: First test

Make it pass:

fragments/search_form.html:

```
<form>
<input class="search-term" type="text">
</form>
```

JavaScript Example: Driver page

```
<html>
<script type="text/javascript"
  src="http://code.jquery.com/jquery-1.11.1.min.js">
</script>
<script type="text/javascript" src="app.js"></script>
<script type="text/javascript">
    function test_edit_box_exists(){ ... }
    $(document).ready(function(){
        if(test_edit_box_exists()){
            alert("Success");
        });
</script>
</html>
```

JavaScript Example: First AJAX test

```
function test_search_sends_ajax(){
  var form=load_search_form(); var posted_ajax=[];
  var dummy_ajax=function(url,data,handler){
    posted_ajax.push({url:url,data:data,handler:handler}
  }
  setup_search_form(form,dummy_ajax);
  form.attr('action','#');
  form.find(input.search-term').val("red widgets");
  form.find('button.search').click();
  if((posted_ajax.length!=1) ||
    (posted_ajax[0].url!='/search') ||
    (posted_ajax[0].data.term!='red widgets')){
    alert('No/Wrong AJAX request posted'); return false;
  }
  return true;
}
```

JavaScript Example: Framework support

Simulating the click is easy:

```
form.find('button.search').click();
```

We don't need to know how the click event is attached to the button.

JavaScript Example: First AJAX test

Make it pass:

app.js:

```
function setup_search_form(form,ajax){
  form.find('button.search').click(function(){
    ajax(
      '/search',
      {term:form.find('input.search_term').val()},
      function(){});
    return false;
  });
}
```

JavaScript Example: Second AJAX test

```
function test_results_go_in_div(){
  var form=load_search_form(); var posted_ajax=[];
  var dummy_ajax=function(url,data,handler){
    posted_ajax.push({url:url,data:data,handler:handler}
  }
  setup_search_form(form,dummy_ajax);
  form.attr('action','#');
  form.find(input.search-term').val("red widgets");
  form.find('button.search').click();
  ...
```

JavaScript Example: Second AJAX test

```
...
var result_data={
  results:[
  "red spinning widgets",
  "fast red widgets",
  "big red widgets"
  ]
};

test_data.posted_ajax[0].handler(result_data);
...
```

JavaScript Example: Second AJAX test

```
    ...
    var result_div=form.find('.results');
    if((result_div.children().length!=1) ||
       (result_div.find('ul').length!=1) ||
       (result_div.find('ul li').length!=3)){
       alert('Results should have a list with one row per
       return false;
    }
    ...
}
```

JavaScript Example: Second AJAX test

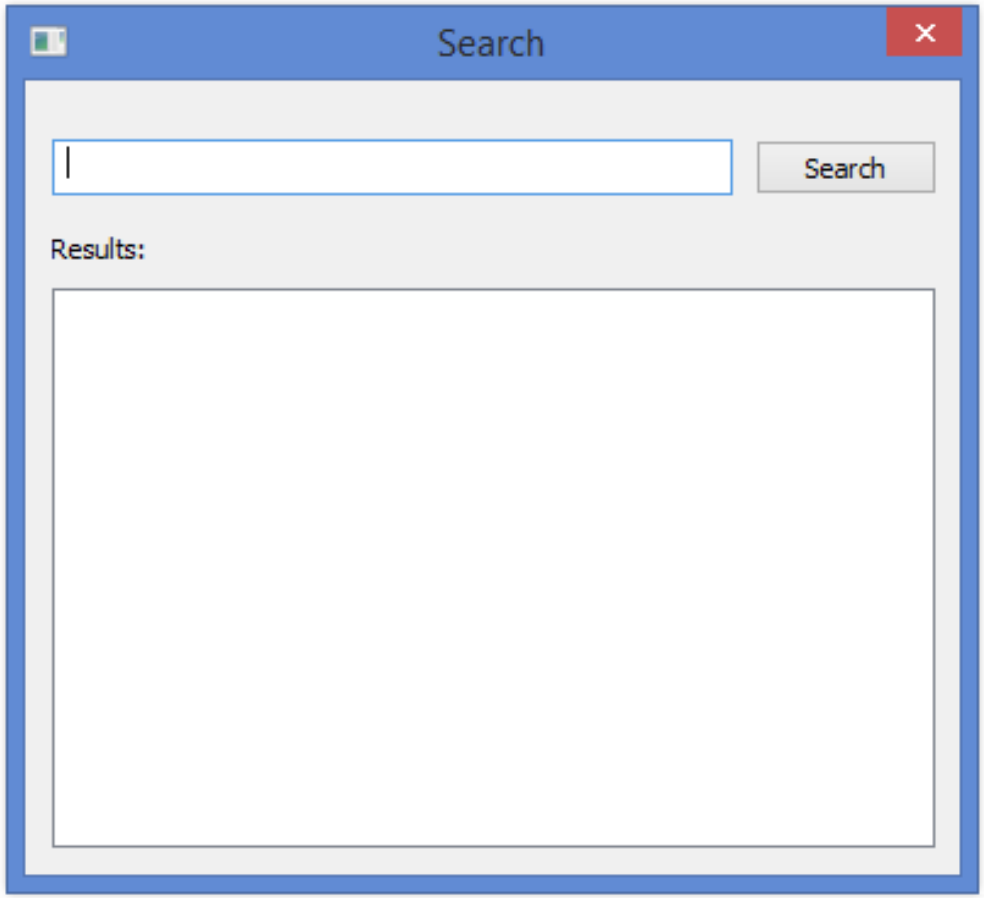
```
function setup_search_form(form,ajax){
  var results_field=form.find('.results');
  var handle_results=function(data){
    var result_list=$('<ul></ul>');
    for(var i=0;i<data.results.length;++i){ ... }
    results_field.append(result_list);
  };
  form.find('button.search').click(function(){
    ajax(
      '/search',
      {term:form.find('input.search_term').val()},
      handle_results);
    return false;
  });
}
```

JavaScript Example: Live AJAX

```
function jquery_ajax(url,data,handler,failure_handler){
  $.post(url,data,handler).fail(function(xhr){
    if(failure_handler){
      failure_handler(
        xhr.status,xhr.statusText,xhr.responseText);
    }
  });
}
```

C++ Example

C++ Example: Search Dialog Mockup



C++ Example: A test

```
void test_search_request_sent_on_click(){
  DummySearchServer server;
  SearchUI display(non_owning_shared_ptr(&server));
  display.create();
  simulateUserTextEntry(
    display,IDC_SEARCH_BOX,"red widgets");
  simulateUserClick(display,IDC_SEARCH_BUTTON);
  assert(server.requests.size()==1);
  assert(server.requests[0].searchTerm=="red widgets");
}
```

C++ Example: the class definition

```
class SearchServer;

class SearchUI: public CDialog{
  shared_ptr<SearchServer> server;
public:
  SearchUI(shared_ptr<SearchServer> server_);
  void create();
protected:
  void OnSearchButtonClicked();
  DECLARE_MESSAGE_MAP();
};
```

C++ Example: the click function

```
void SearchUI::OnSearchButtonClicked(){
  CString s;
  GetDlgItemText(IDC_SEARCH_BOX,s);
  server->search(s);
}

BEGIN_MESSAGE_MAP(SearchUI,CDialog)
ON_COMMAND(IDC_SEARCH_BUTTON,OnSearchButtonClicked)
END_MESSAGE_MAP()
```

C++ Example: test support functions

C++ frameworks often don't make it easy to simulate user interactions

C++ Example: test support functions

C++ frameworks often don't make it easy to simulate user interactions

...so we have to write our own support functions (`simulateUserTextEntry` and `simulateUserClick`).

C++ Example: test support functions

On Windows, you have a choice:

1. Use the UI Automation API to simulate the user interactions,
2. Use the `SendInput` API to simulate the input to the control, or
3. Update the controls and send the messages that would be generated

I prefer option 3.

C++ Example: test support functions

```
void simulateUserTextEntry(
  CWnd& window,int controlId,CString text){
    auto control=window.GetDlgItem(controlId);
    assert(control);
    assert(getClassName(control)==WC_EDIT);
    assert(control->IsWindowEnabled());
    auto editCtrl=(CEdit*)control;
    if(text.GetLength()>editCtrl->GetLimitText())
      text=text.Left(editCtrl->GetLimitText());
    SetWindowText(editCtrl->m_hWnd, text);
    auto parent=editCtrl->GetParent(); assert(parent);
    parent->SendMessage(
      WM_COMMAND,MAKELONG(controlId,EN_CHANGE),
      (LPARAM)editCtrl->m_hWnd);
}
```

C++ Example: test support functions

You need a support function for each type of user interaction:

- Button clicks,
- Drop-down list selection,
- Right clicks,
- Menu selection
- ...

Guidelines

Guidelines

Mock out everything that isn't the focus of the test:

- Database access
- File access
- Other UI components such as message boxes

Guidelines

Trust the framework:

- Query controls for their data/settings
- Wrap controls that don't allow queries
- Do not test the pixels unless you're testing a graphical control

Guidelines

Write a library of support functions:

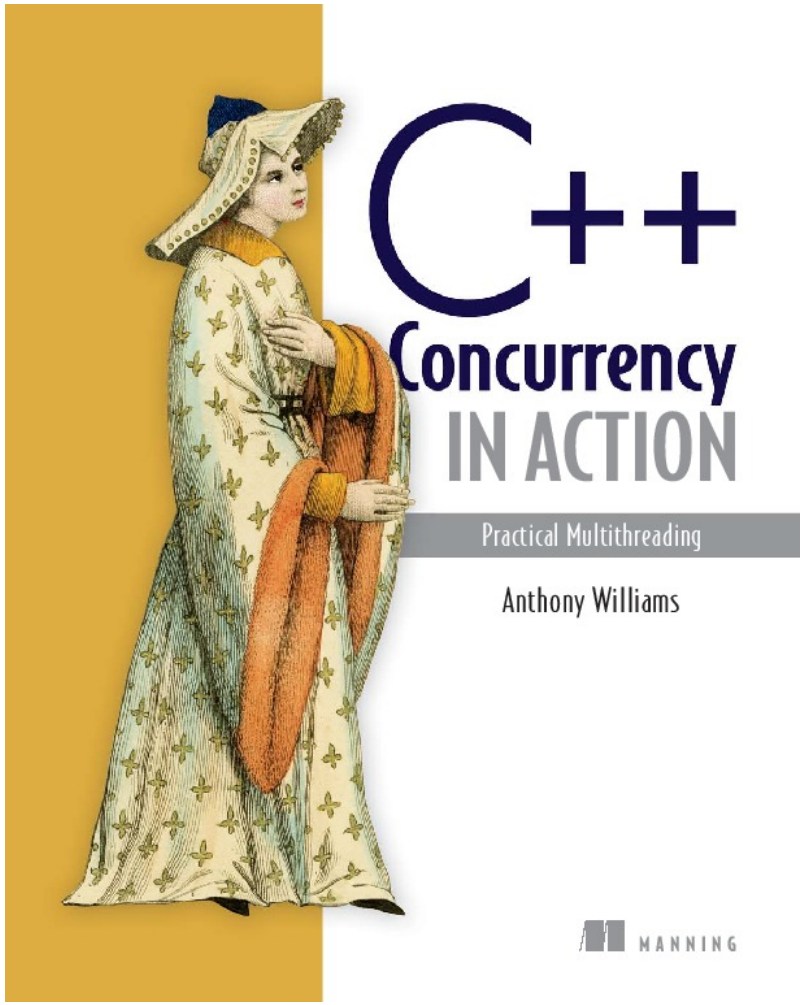
- Write each function as you need it
- Use the framework where you can

Guidelines

- Red-green-refactor
- Baby steps — Do The Simplest Thing That Could Possibly Work
- Have fun!

Questions?

My Book



C++ Concurrency in Action: Practical Multithreading

`http://stdthread.com/book`

Picture credits

The images listed below are from the specified source, with the specified license. All other images are copyright Just Software Solutions Ltd, licensed under Creative Commons Attribution ShareAlike 4 https://creativecommons.org/licenses/by-sa/4.0/.

- Penguin Questioning: https://commons.wikimedia.org/wiki/File:Penguin_questioning_by_mimooh.svg by Mimooh, Creative Commons Attribution-Share Alike 3.0 Unported
- Seams: https://pixabay.com/en/fabric-blue-jeans-thread-seam-676564/ by Alexei Abramov, Public Domain
- Success sign: https://commons.wikimedia.org/wiki/File:Success_sign.jpg by Keith Ramsey (RambergMediaImages), Creative Commons Attribution ShareAlike
- Difficult Problems: https://www.flickr.com/photos/31733144@N04/3131350666 by wakefielddavid, Creative Commons Attribution 2.0