Ouverture

Avere uno strumento per tradurre casi d'uso in test di integrazione garantisce:

- Un interpretazione uniforme del significato di casi d'uso per tutti i test del sistema.
- Che si prova tutti i (documentate) casi d'uso.

Presenterà una semplice implementazione di un tale strumento, e discuterò le mie esperienze con l'utilizzo su un progetto di sviluppo di software commerciale (ma Open Source). Concludo con alcune idee su come lo strumento potrebbe essere migliorata (riscritta).

Jacob Sparre Andersen
+45 21 49 08 04

Generating Integration Tests from Use-cases

Jacob Sparre Andersen

JSA Research & Innovation

Driving IT 2015, IDA, Copenhagen

- Not really about **a** tool.
- Not really about a process either.

It is – in a way – about a feature you might want to add to your existing development process.

- It even works for the waterfall process.

Jacob Sparre Andersen
+45 21 49 08 04

1. You want to test a software system.
2. You have use-cases (scenarios) describing how the software system is intended to behave. (In a language the user understands.)
3. It has a value to be able to document which use-cases have been implemented correctly.
4. The use-cases may be changed after you run your first test.

Jacob Sparre Andersen
+45 21 49 08 04

Given the assumptions:

- It makes sense to have a tool which automatically, consistently (and hopefully correctly) translates/compiles your use-cases to integration tests.

Because:

- The tool gives us trivial traceability from use-cases to test results.
- The tool allows a continuous integration system to reflect changes to use-cases as soon as they have been committed.
- The tool does the work of adjusting a test, when the corresponding use-case is modified.

Jacob Sparre Andersen
+45 21 49 08 04

A general tool of this kind can't have any domain knowledge or know enough about the interfaces to the software system to generate the integration tests completely on its own.

The solution to this is to map the use-cases to source files, calling operations named in the domain terminology, and then having the test developers make sure all the required operations are available with suitable implementations.

- So we still have to *write* the tests, just not everything, and only in small bites.

Jacob Sparre Andersen
+45 21 49 08 04

Some tools

- Cucumber (`https://cucumber.io/`)
- "Iteration 1" (`https://github.com/AdaHeads/Hosted-Telephone-Reception-System`)
- KIT-g (`http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6900`)

(in chronological order)

Jacob Sparre Andersen
+45 21 49 08 04

Cucumber

- Translates using patterns in the operations implementing the domain knowledge.
- Mature.

But:

- Not supporting The Right Languages(TM).
- NIH.

Jacob Sparre Andersen
+45 21 49 08 04

Cucumber (example)

Example:

```
Given I have entered 50 into the calculator
```

is matched by

```
Given /I have entered (.*) into the calculator/ do |n|
  calculator = Calculator.new
  calculator.push(n.to_i)
end
```

Jacob Sparre Andersen
+45 21 49 08 04

"Iteration 1"

- In-house experimental tool at AdaHeads.
- Contains mappings of lines in the use-cases to lines in the integration test source files.
- Supports The Right Languages(TM).

But:

- Having to maintain explicit mappings is time consuming. (And annoying.)
- Limited language support (Python, Ada).
- Immature.

Jacob Sparre Andersen
+45 21 49 08 04

KIT-g

- Uses tagging of *agents* and *concepts* in the use-cases to allow automatic mapping of lines in the use-cases to lines in the integration test source files.

But:

- Limited language support (Dart).
- The tagging needs more work.
- Immature.

Jacob Sparre Andersen
+45 21 49 08 04

KIT-g (example)

Example:

```
Receptionist types in message
```

may be tagged as:

```
*Receptionist* types in *message*
```

(assuming that *Receptionist* is on the agent list and *message* is on the concept list)

The mapping is then:

```
_receptionist_types_in_message (receptionist, message);
```

+45 21 49 08 04

- Tools may help you maintain your test sources.

Comments?

Jacob Sparre Andersen
+45 21 49 08 04

Informazioni di contatto e links

Jacob Sparre Andersen
JSA Research & Innovation
jacob@jacob-sparre.dk
http://www.jacob-sparre.dk/

I miei archivi software Open Source:
http://repositories.jacob-sparre.dk/