Systematic monkey and how fMBT helps implementing it

Antti Kervinen

Intel

antti.kervinen@intel.com

Software testing day, Tampere
June 6th, 2012

Systematic monkey

"If a monkey keeps randomly typing infinitely long time, it almost surely will write any given text."

Systematic monkey

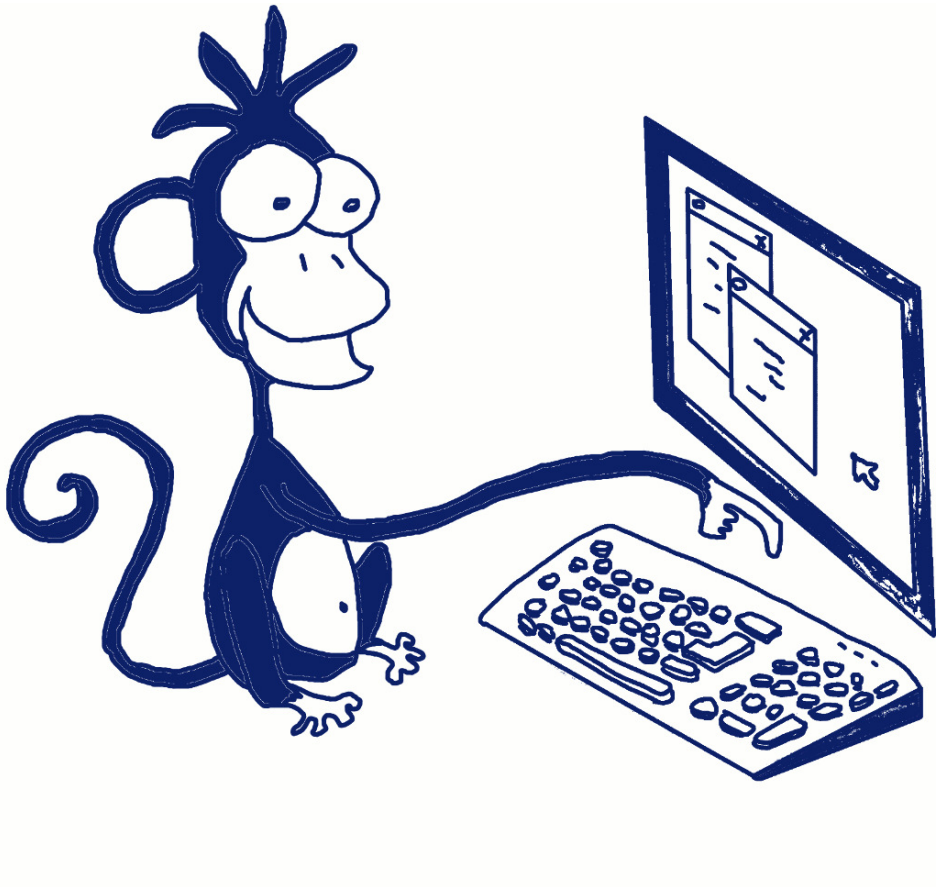
> *"If a monkey keeps randomly typing infinitely long time, it almost surely will write any given text."*

Monkey testing (stochastic testing) is automatic random testing.

- Computer acts like a "monkey" giving random inputs.
- The longer the test runs, the more likely it will give inputs that break the system.

Systematic monkey

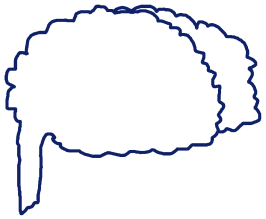
How to do monkey testing in practice?



Systematic monkey

How to do monkey testing in practice?

We'll need...

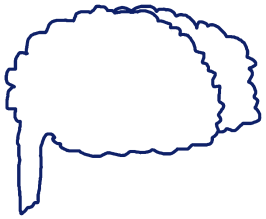


logic

Systematic monkey

How to do monkey testing in practice?

We'll need...

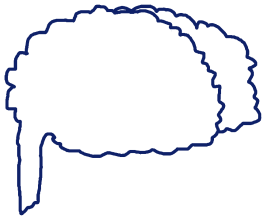


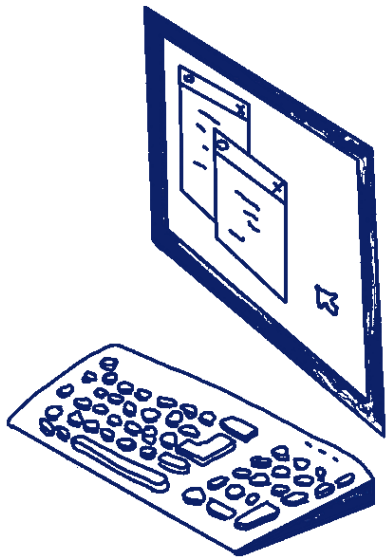
logic, adaptation

Systematic monkey

How to do monkey testing in practice?

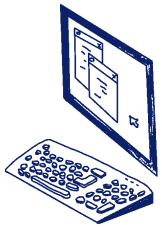
We'll need...





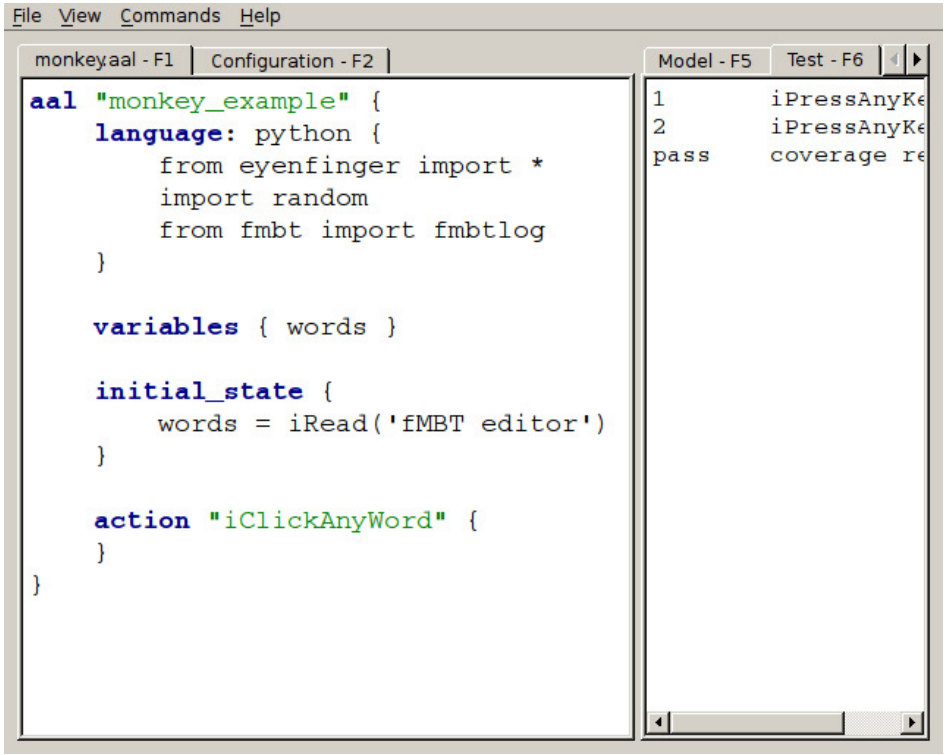
logic, adaptation and something to test.

Systematic monkey



Let's take an example. First, something to test.

We'll test this kind of GUI software:



It suffices to know that one can type on it and click on it here and there.

Systematic monkey

Let's consider the adaptation next.

We'll use fMBT's `eyenfinger` library:

`iRead()`	The eye reads all words on the display.
`iClickWord()`	The finger clicks given word using mouse.
`iType()`	The finger hits given keys on the keyboard.
`iVerifyWord()`	Library verifies if given word resembles any of read words.

This is all in place, we've done nothing yet.

Systematic monkey

Finally, the logic.

We'll write it in fMBT's AAL language. The structure is

Action 1: if condition 1, can do instructions 1.
Action 2: if condition 2, can do instructions 2.
...

Systematic monkey

Finally, the logic.

We'll write it in fMBT's AAL language. The structure is

Action 1: if condition 1, can do instructions 1.
Action 2: if condition 2, can do instructions 2.
...

For instance,

- Action: Click any word.
- Condition: There's at least one word on the display
- Instructions:
  1. Randomly pick a word on the display.
  2. Click the word.
  3. Re-read the contents of the display.

Systematic monkey

The same in AAL/Python using the eyenfinger library:

```
action "iClickAnyWord" {
    guard() {
        return words != []
    }
    adapter() {
        word = random.choice(words)
        iClickWord(word)
        words = iRead()
    }
}
```

Systematic monkey

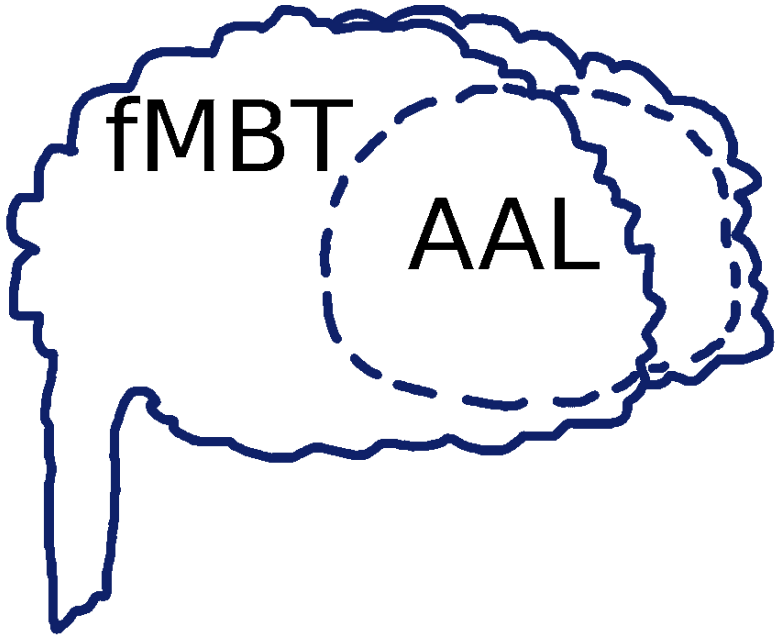
The same in AAL/Python using the eyenfinger library:

```
action "iClickAnyWord" {
    guard() {
        return words != []
    }
    adapter() {
        word = random.choice(words)
        iClickWord(word)
        words = iRead()
    }
}
```

This is all the logic that we need to implement a simple monkey. This is enough, because...

Systematic monkey

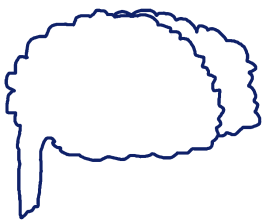
. . . fMBT will take care of the rest.



fMBT finds out which of all the actions are enabled and makes the decision which to test next.

Systematic monkey

This is where we are now:

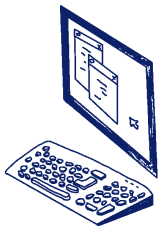


fMBT & AAL logic

+

eyenfinger adaptation

+

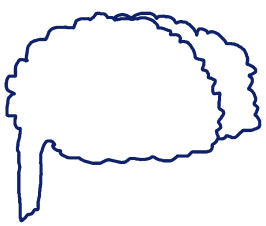


GUI to test

=

Systematic monkey

This is where we are now:



fMBT & AAL

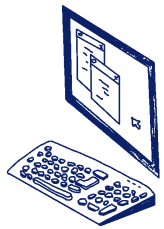
logic

+

eyenfinger

adaptation

+



GUI

to test

=

But could we make this monkey any smarter? Could it “understand” even a bit what the software could do?

Systematic monkey

We can add more logic to monkey's brain in AAL. For instance,

Systematic monkey

We can add more logic to monkey's brain in AAL. For instance,

- Action: Open a file.

Systematic monkey

We can add more logic to monkey's brain in AAL. For instance,

- Action: Open a file.
- Condition: There's no "Open" word on the display.

Systematic monkey

We can add more logic to monkey's brain in AAL. For instance,

- Action: Open a file.
- Condition: There's no "Open" word on the display.
- Instructions:
  1. Click word "File".
  2. Read words again.
  3. Click word "Open".
  4. Type any of the following: "monkey.aal", "emptyfile.aal", "unreadablefile.aal" or "readonlyfile.aal".
  5. Press Return.
  6. Read words again.
  7. If "monkey.aal" was chosen, verify that there is "monkey_example" on the display.

Now our monkey will test clicking random words on the display *and* opening any of the three files at any point – unless the File menu or Open file dialog is already open due to random clicking.

Systematic monkey

What have we seen so far?

Conditions on previous action samples depend *only* on what monkey sees on the display.

Systematic monkey

What have we seen so far?

Conditions on previous action samples depend *only* on what monkey sees on the display.

- If there's at least one word on the display, testing "Click any word" is enabled.
- If there's no word "Open" on the display, testing "Open a file" is enabled.

Systematic monkey

What have we seen so far?

Conditions on previous action samples depend *only* on what monkey sees on the display.

- If there's at least one word on the display, testing "Click any word" is enabled.
- If there's no word "Open" on the display, testing "Open a file" is enabled.

What if conditions would depend *only* on what monkey understands it has done?

Systematic monkey

What have we seen so far?

Conditions on previous action samples depend *only* on what monkey sees on the display.

- If there's at least one word on the display, testing "Click any word" is enabled.
- If there's no word "Open" on the display, testing "Open a file" is enabled.

What if conditions would depend *only* on what monkey understands it has done?

- If a read-only file has been opened, testing "error in saving" is enabled.

Systematic monkey

What have we seen so far?

Conditions on previous action samples depend *only* on what monkey sees on the display.

- If there's at least one word on the display, testing "Click any word" is enabled.
- If there's no word "Open" on the display, testing "Open a file" is enabled.

What if conditions would depend *only* on what monkey understands it has done?

- If a read-only file has been opened, testing "error in saving" is enabled.

This would be **model-based testing**! Unlike pure monkey testing, it's suitable for testing requirements, too.

Systematic monkey

What have we seen so far?

Conditions on previous action samples depend *only* on what monkey sees on the display.

- If there's at least one word on the display, testing "Click any word" is enabled.
- If there's no word "Open" on the display, testing "Open a file" is enabled.

What if conditions would depend *only* on what monkey understands it has done?

- If a read-only file has been opened, testing "error in saving" is enabled.

This would be **model-based testing**! Unlike pure monkey testing, it's suitable for testing requirements, too.

AAL enables combining monkey testing and model-based testing!

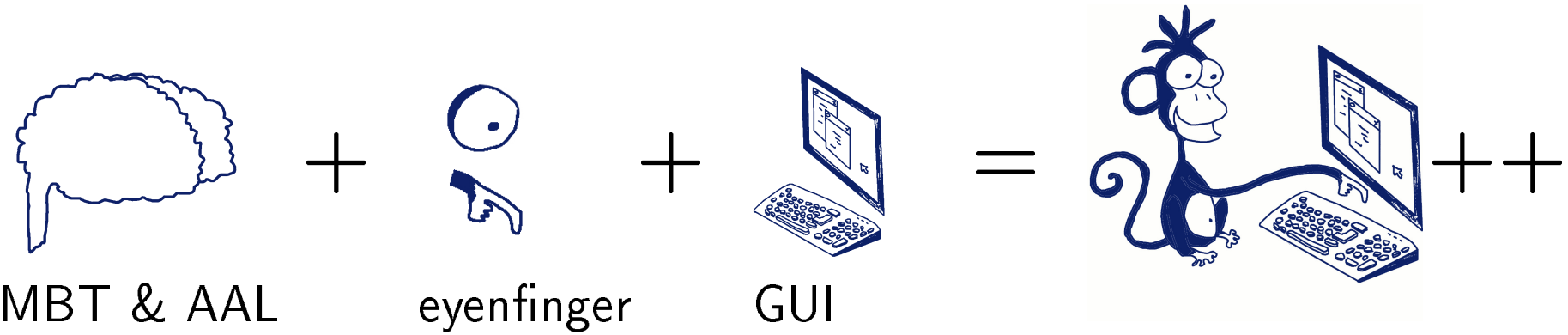
Systematic monkey

After words:

- Dumb monkeys are extremely easy to implement, but they can't test requirements and don't recognize most errors.
- Model-based testing requires more effort, tests requirements and detects errors, but leaves unexpected behaviors untested.
- In AAL actions can depend on from *both* observations on the real system *and* understanding what is testable.

⇒ With AAL one can do monkey testing, model-based testing and what falls between these two.

- fMBT tests enabled actions in both random and systematic ways.



fMBT & AAL eyenfinger GUI

Systematic monkey

- fMBT (including eyenfinger) is available at http://github.com/pablovirolainen/fMBT
- Open source, LGPL license