Making my macros, scripts and plugins better by refactoring

I2K 2022

Montpellier Ressources Imagerie
Leo Tellez and Volker Bäcker

Introduction

- “Refactoring: a change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behavior.”

    Refactoring: Improving the Design of Existing Code by Martin Fowler, Kent Beck, John Brant, William Opdyke, Don Roberts, 1999, Addison-Wesley Professional, ISBN: 0201485672

Introduction

- Refactor
  - When you have to add a feature to a program
    - and the code is not structured in a convenient way to add the feature
  - When you get a bug report, start by writing a unit test that exposes the bug.
  - When you feel the need to write a comment
- Before you start refactoring, check that you have a solid suite of tests.
- Refactoring changes the programs in small steps.
- Three strikes and you refactor.

“Any fool can write code that a computer can understand. Good programmers write code that humans can understand.” Martin Fowler

Why should you refactor?

- Refactoring Improves the Design of Software
- Refactoring Makes Software Easier to Understand
- Refactoring Helps You Find Bugs
- Refactoring Helps You Program Faster

Code smells (examples)

Code smell	Refactorings
Duplicated Code	Extract Method, Extract Class, ...
Long Method	Extract Method, Method Object, ...
Lots of Parameters	Parameter Object, ...
Large Class	Extract Class, Extract Subclass, ...
Speculative Generality	Remove Parameter, Rename Method, ...

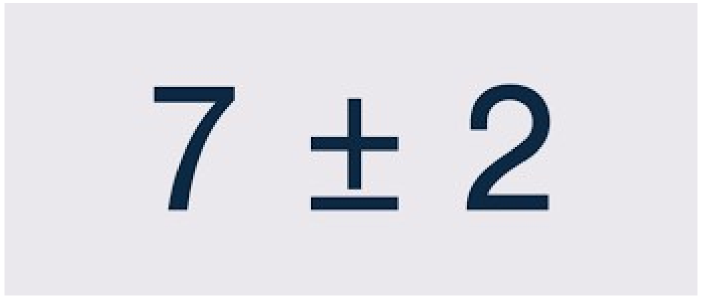
What is long, large ?

7 ± 2

But, no more than
3 positional parameters
3! = 6, 4! = 24

Software engineering principles

- Some basic principles to start with:
  - break your code down in small units
    - each unit does one thing (has one responsibility)
    - each unit has no more than 7±2 subunits
    - consistent level of abstraction
  - control redundancy
    - never have the same 2+ lines of code appear multiple times
    - never use literal values in the middle of the code
  - write code for the human reader
    - use self-explaining names
    - avoid abbreviations that are not domain standard
      - stdDev is ok, imp is not
  - avoid clutter
    - prefixex, type, scope, ...



Programming is Gardening, not Engineering

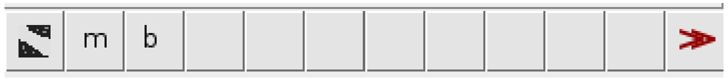
- The garden doesn't quite come up the way you drew the picture.
- This plant gets a lot bigger than you thought it would.
  - You've got to prune it.
  - You've got to split it.
  - You've got to move it around the garden.
- This big plant in the back died.
  - You've got to dig it up and throw it into the compost pile.
- These colors ended up not looking … good next to each other
  - You've got to transplant this one over to the other side of the garden.

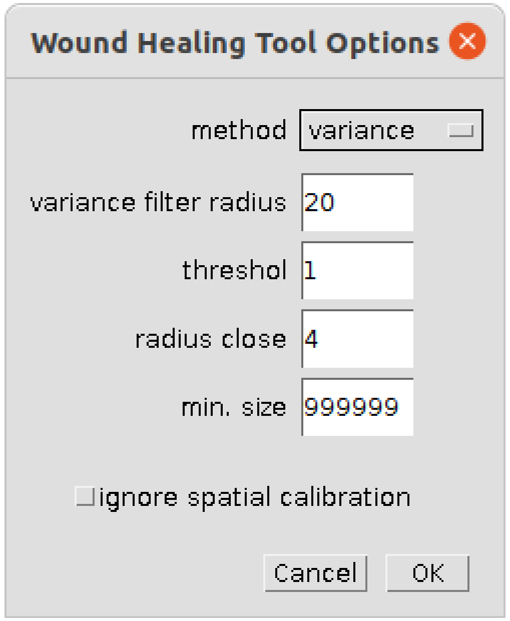
Andy Hunt

The Pragmatic Programmer, Andrew Hunt and David Thomas, 1999, Addison Wesley, ISBN 0-201-61622-X

https://www.artima.com/articles/programming-is-gardening-not-engineering

Example




Wound Healing Tool Options

method: variance

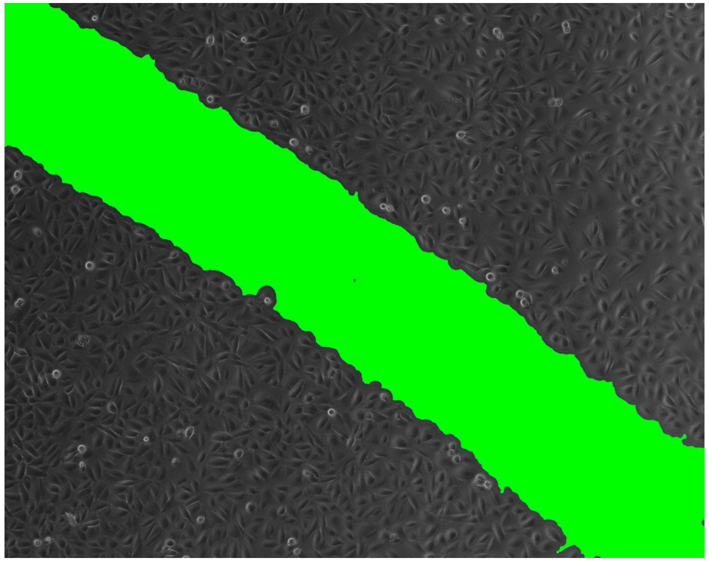
variance filter radius: 20

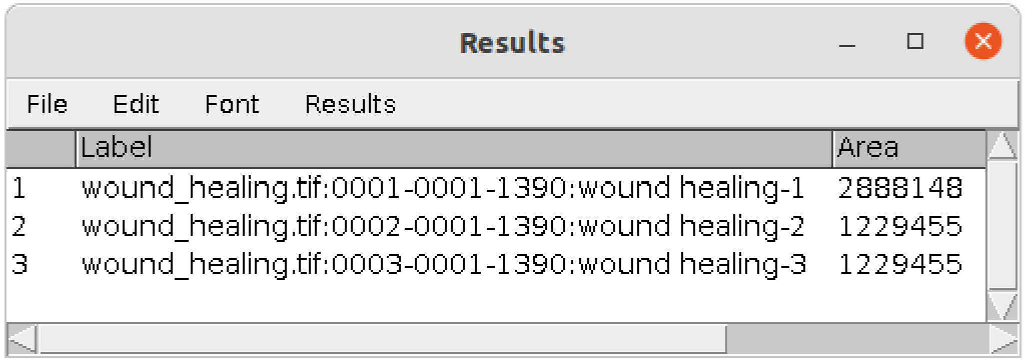
threshol: 1

radius close: 4

min. size: 999999

☐ ignore spatial calibration




Results

	Label	Area
1	wound_healing.tif:0001-0001-1390:wound healing-1	2888148
2	wound_healing.tif:0002-0001-1390:wound healing-2	1229455
3	wound_healing.tif:0003-0001-1390:wound healing-3	1229455

Step one

- Write unit tests

```
function testIsInputImage() {
    image1 = "test.tif";
    result = isInputImage(image1);
    image2 = "test.TIF";
    result = result && isInputImage(image2);
    image3 = "test.png";
    result = result && !isInputImage(image3);
    return result;
}
```

Step two

- Make the global variables visible
  - varianceFilterRadius
  - VARIANCE_FILTER_RADIUS
- So that the difference between local and global variables becomes evident
- Find and rename, but check with Find first

```
if (MEASURE_IN_PIXEL_UNITS)
        run("Set Scale...", "distance=0 known=0 pixel=1 unit=pixel");
getPixelSize(unit, pixelWidth, pixelHeight);
```

Step three

- Remove unused code

```
if (MEASURE_IN_PIXEL_UNITS)
        run("Set Scale...", "distance=0 known=0 pixel=1 unit=pixel");
getPixelSize(unit, pixelWidth, pixelHeight);
```

- Find reveals that unit, pixelWidth and pixelHeight are never used

```
if (MEASURE_IN_PIXEL_UNITS)
        run("Set Scale...", "distance=0 known=0 pixel=1 unit=pixel");
```

Step four

- Replace algorithms by “better”, equivalent algorithms
  - Make sure the results are absolutely identic including side effects

```
for (i=0; i<radiusOpen; i++) {
    run("Dilate", "stack");
}
for (i=0; i<radiusOpen; i++) {
    run("Erode", "stack");
}
```

```
run("Options...", "iterations="+RADIUS_CLOSE+" count=1 black do=Close stack");
run("Options...", "iterations=1 count=1 black do=Nothing");
```

Step Four part 2

```
run("Options...", "iterations="+RADIUS_CLOSE+" count=1 black do=Close stack");
run("Options...", "iterations=1 count=1 black do=Nothing");
run("Select All");
run("Enlarge...", "enlarge=-" + RADIUS_CLOSE + " pixel");
```

- Using pad makes the correction of the border unnecessary
- This changes behavior, but fixes a bug

```
run("Options...", "iterations="+RADIUS_CLOSE+" count=1 pad black do=Close stack");
run("Options...", "iterations=1 count=1 black do=Nothing");
```

Interlude

```
run("Select None");
if (MEASURE_IN_PIXEL_UNITS)
run("Set Scale...", "distance=0 known=0 pixel=1 unit=pixel");
run("Duplicate...", "duplicate");
setForegroundColor(0, 0, 0);
setBackgroundColor(255, 255, 255);
roiManager("reset")
roiManager("Associate", "true");
```

→ Some setup

```
if (METHOD=="variance")
    thresholdVariance();
else
    thresholdFindEdges();
run("Convert to Mask", " black");
resetThreshold();
```

← Create mask, gap foreground

Interlude

Morphological close on tissue

```
run("Invert", "stack");
run("Options...", "iterations="+RADIUS_CLOSE+" count=1 pad black do=Close stack");
run("Options...", "iterations=1 count=1 black do=Nothing");
run("Invert", "stack");
```

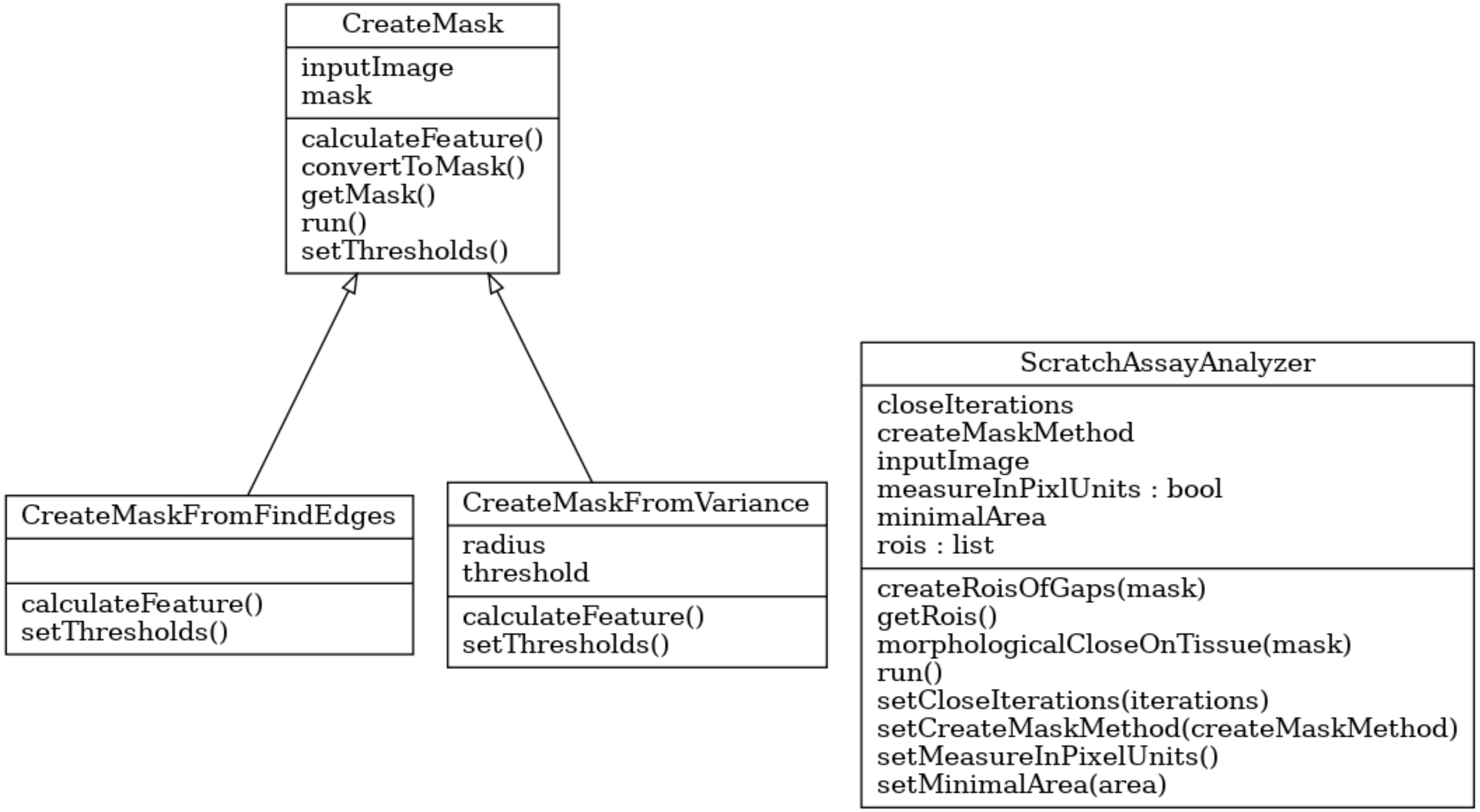
```
run("Analyze Particles...", "size="+MINIMAL_SIZE+"-Infinity circularity=0.00-1.00
show=Nothing add stack");
close();
run("Clear Results");
roiManager("Measure");
roiManager("Show None");
roiManager("Show All");
```

Step five

- Extract functions

```
function measureActiveImage() {
    if (MEASURE_IN_PIXEL_UNITS) removeScale;
    initialize();
    createMaskWithGapAsForeground(METHOD, VARIANCE_FILTER_RADIUS, THRESHOLD);
    applyMorphologicalCloseOnTissue(RADIUS_CLOSE);
    createRoisOfGaps(MINIMAL_SIZE);
    closeMask();
    roiManager("Measure");
    roiManager("Show All");
}
```

OOP version in jython



```
def main():
    analyzer = getAnalyzer()
    analyzer.run()

def getAnalyzer():
    analyzer = ScratchAssayAnalyzer(inputImage)
    if measureInPixelUnits:
        analyzer.setMeasureInPixelUnits()
    createMaskMethods = {"variance"   : CreateMaskFromVariance(inputImage, filterRadius, threshold),
                         "find edges" : CreateMaskFromFindEdges(inputImage)}
    analyzer.setCreateMaskMethod(createMaskMethods[feature])
    analyzer.setCloseIterations(closeRadius)
    analyzer.setMinimalArea(minimalArea)
    return analyzer

main()
```

Let’s get to work