Seb Rose - Rational Jazz Server
Jazz is an IBM initiative to help make software delivery teams more effective
  • an architecture for lifecycle integration
  • a portfolio of products designed to put the team first
  • a community of stakeholders - [http://jazz.net](http://jazz.net)

Open Services for Lifecycle Collaboration (OSLC) to promote vendor interoperability
  • [http://open-services.net](http://open-services.net)

JAZZ Integration Architecture (JIA)
  • defines a set of cross-tool services (JFS), extended by products
  • all services exposed via RESTful interface
  • tools separated from definition & access to data

Rational JAZZ Team Server (JTS)
  • Implements the JFS
Didier Verna - Lisp, Jazz, Aikido
LISP, Jazz, Aïkido
Three tales of the same story

Didier Verna

April 23, 2009
My philosophy of life

- **Beauty:** being able to evolve comfortably within a set of constraints, which begins with accepting their existence.

- **Fun:** being able to break those constraints at will, and then going back to them at will.

- **Unification:** drawing bridges between *(a priori)* unrelated domains, in the search for the essence of all things.
Where is the beauty?

- **LISP** Writing code in any language can be beautiful, provided you know how to adapt your own ideas to the language’s constraints.

- **Jazz** Playing a song in any kind of music can be beautiful, provided you know how to adapt your own ideas to the music’s constraints.
Where is the fun?

- **Lisp**  You can adapt the language to your own ideas instead of just having to adapt your own ideas to it, hence effectively breaking the rules or ordinary languages (“programmable programming language”).

- **Jazz** Improvisation (the essence of Jazz), lets you adapt the music to your own ideas instead of forcing you to adapt your ideas to it, effectively breaking the rules of ordinary music.
Unification

- **LISP** Because it’s a meta-language, LISP is imperative, procedural, functional, object-oriented, declarative, anything you want.

- **Jazz** Because Jazz is not a kind of music, but a way to address all kinds of music, in Jazz there is jazz, classical, pop, rock, hard-rock, rap, electro, anything you want.
Conclusion

The Tao

(setq *lisp* (make-instance 'my-phi-of-life))
(setq *jazz* (make-instance 'my-phi-of-life))

- Do you love Jazz?
- Then, you should program in LISP.
- Otherwise, you are **wrong**.

- Do you already program in LISP?
- Then, ...
Perspectives

- Rush to http://www.didierverna.com
- Buy my CD !!

Available on:
- iTunes
- CDBaby
- Amazon
- Napster
- ...

Related blog

Mark Bartosik - Hunting For Bugs
Golden rules

• You are guilty until proven innocent
• Always generate debugging symbols
• The symbol server is your best friend
• All access violations are deadly
• Save a .DMP file
• Have sharp tools www.sysinternals.com and “Debugging Tools for Windows”, VS2005
Using WER
(example crash)

• WER will report Shell detected hangs “Application not responding”
• WER will report unhandled exceptions
Using WER
Reading .dmp files with WinDebug

• First download the latest Debugging Tools for Windows package. It is updated about twice a year, plus beta releases, currently at v6.6.

• Set the symbol path:
  .sympath
  or from the menu
  File, Symbol File Path…

• Make sure the Microsoft symbol server is on the path:
  Add SRV*c:\websymbols*http://msdl.microsoft.com/download/symbols
  or use
  .symfix+ c:\websymbols

• Make sure that path to the binary images is set
  .exepath+ c:\where_your_binaries_are
  or from the menu
  File, Image File Path…

• File, Open Crash Dump… (older versions also Debug, Go)
Capturing post mortem files programmatically

• Implement a vectored exception handler, not simple, catches almost everything.
• Implement an unhandled exception filter, easy but does not catch everything.
• Implement an se_translator function easy but not suitable for all projects.
• Implement an exception filter with __except, can be messy, scoped, easy for per thread, does not catch everything.
• Implement catch(...), not recommended.
• Whatever the mechanism used to intercept exceptions, we need a process to create the dumpfile (doing this in-proc is not recommended).
• A typical implementation will CreateProcess with a command line of “dumpcapture-program -p pid”.
Debugging exceptions

• Always trap access violations

• First column is *first chance* second column is *second chance* (unhandled)
gflags

Part of Debugging Tools for Windows

Do not enable both heap tail checking and page heap
Golden rules

• You are guilty until proven innocent
• Always generate debugging symbols
• The symbol server is your best friend
• All access violations are deadly
• Save a .DMP file
• Have sharp tools www.sysinternals.com and “Debugging Tools for Windows”, VS2005
Jim Hague - A Thought For St George's Day
If it were done when 'tis done, then 'twere well
It were done quickly

- Macbeth, Act I Scene VII
Beware the ides of March.

- Julius Caesar, Act I Scene II
Beware the march of IDEs.
• Build system
• Version control view warping
• Coding by fiddling
• Fixing you into one window
Kevlin Henney - The Uncertainty Principle
Charles Bailey - git log --summary @{1 year ago} @{now}
Who am I?

Charles Bailey

charles.bailey@igence.com
git log --summary @{1 year ago} @{now}
How many?

- v1.5.4.5 to v1.6.2.1
- 3410 non-merge commits
- 910 merges
It works

- Built on a simple system that works
- The object model has not changed since 1.0
- It's unlikely to change
Staging area

git stage <files>

(aka: git add)
Stage some changes...

git stage -p
Don't lose your HEAD

- `git reflog`
- `git reset HEAD@{n}`
Don't lose your changes

`git stash`
Status isn't

git status =~ git commit --dry-run --verbose
rebase: update patch to latest

\[ \text{O} \quad \text{O} \quad \text{A} \quad \text{O} \quad \text{O} \quad \text{B} \]

\[ \text{\backslash} \quad \text{O} \quad \text{C} \quad \text{<- I'm here} \]

git rebase B

\[ \text{O} \quad \text{O} \quad \text{A} \quad \text{O} \quad \text{O} \quad \text{B} \quad \text{O'} \quad \text{C'} \quad \text{<- I'm here} \]

\[ \text{\backslash} \quad \text{O} \quad \text{C} \]

rebase: move patches to latest

O – O – A – O – O – B
\ – O – C – O – D  <- I'm here

git rebase --onto B C
O – O – A – O – O – B – O' – D' <- I'm here
\ – O – C – O – D ]
Olve Maudal - Analogy: A Codebase Is Like A Kitchen
Your codebase is like a kitchen

Olve Maudal
oma@pvv.org

5 minute lightening talk at ACCU 2009
April 24, 2009
The only valid measurement of code quality: WTFs/minute

Good code.

Bad code.
The state of your codebase determines what you can achieve
Analogy

The codebase is like a kitchen
suppose you are just going to make something nice for yourself
then, really, anything will do.

Even...
but, software development is usually about more than just making something nice for yourself.

It is usually about making something really fancy...
壽司

**sushi**

which one do you like best?
together with a large team of professionals...
for some demanding customer...
Then it is obvious:

To succeed you need a clean and functioning working environment.
Your codebase is like a kitchen.

Keep it clean so that you can create spectacular solutions for your demanding customers!
The Boy Scout Rule

Leave the campground cleaner than you found it.
The Importance of hygiene

Uncle Bob has suggested that we are now about to "discover" techniques and principles in software engineering that can be compared to the discovery of the importance of hygiene in hospitals by Ignaz Semmelweis in the middle of the 19th century.

Semmelweis found that by introducing hand washing standards before surgery the number of fatal incidence caused by diseases dropped drastically. At the time, diseases were attributed to many different unrelated causes. Each case was considered unique, just like a human person is unique. Semmelweis' hypothesis, that there was only one cause, that all that mattered was cleanliness, was extreme at the time, and was largely ignored, rejected or ridiculed. [wikipedia]

At the point where a certain standard hygiene was accepted and enforced by the medical establishment, doctors started to behave as a group of professionals. This happened about 60 years after Semmelweis' discovery. As software engineers we are not always behaving like professionals, especially not at times where we let pressure from management and customers decide whether we write clean code or not. We know that dirty code is going to slow us down and delay the project, but still, for some reason, we sometimes end up in situations where we do exactly what we are not supposed to do. Imagine how a group of doctors today would react to a situation where they are told not to wash their hands between surgeries? Doctors act as professionals. Unfortunately, as a group of software engineers, we are not there... yet.
Stewart Brodie - Character Encodings: Decoded & Demystified
Universal Character Set (UCS)

All (UCS-4) characters are identified by a 21-bit number terminology: “code point”

All characters are also identified by a name

Example: 'Latin Capital Letter A' written as: U+0041

“Universal” means nearly universal

U+1F04E is …
Universal Character Set (UCS)

All (UCS-4) characters are identified by a 21-bit number terminology: “code point”

All characters are also identified by a name

Example: 'Latin Capital Letter A' written as: U+0041

“Universal” means nearly universal … no Klingon though!

U+1F04E is …
Universal Character Set (UCS)

All (UCS-4) characters are identified by a 21-bit number terminology: “code point”

All characters are also identified by a name

Example: 'Latin Capital Letter A' written as: U+0041

“Universal” means nearly universal … no Klingon though!

U+1F04E is …
UCS-4: Pros & cons

Pros:

Code points storable in 32-bit integers
Very easy and fast to process in code

Cons:

Wasteful of memory: 4 bytes per character!
(Of course, given enough memory & disc space ... )
Encodings are ...

... a mapping for converting a sequence of bytes to a sequence of code points (and vice versa)

... not always able to represent every code point
e.g. ASCII can only represent code points U+0000-U+007F

... sometimes fixed length, e.g. ISO Latin 1 (ISO 8859-1)

... sometimes variable length, e.g. UTF-8

... sometimes stateful! e.g. ISO/IEC 2022
UTF-32: four-byte based encoding (≈ UCS-4)
UTF-16: two-byte based encoding (≈ UCS-2)
(with different endianness variants!)

UTF-8: single byte based, variable-length encoding
- Invented by Ken Thompson (Bell Labs) 1992
- All basic Latin characters encoded in 1 byte
- Most other European characters in 2 bytes
- Most Far Eastern characters in 3 bytes
- All Basic Multi-lingual Plane in 1, 2 or 3 bytes
- Many useful properties (re-sync, error detect, bi-di iteration)
- Can be processed (mostly) safely by usual C string APIs
Using the wrong encoding

“The project cost Â£1M”
“The project cost Â£1M”

Indicative of using Latin 1 to decode UTF-8 byte stream

UTF-8 encoding for U+00A3 is: 0xC2 0xA3

Latin 1 encoding for U+00C2 (Ã) is: 0xC2
Latin 1 encoding for U+00A3 (£) is: 0xA3

Need to use the correct encoding to get the message right!
April 24th is a special day

Happy Birthday, Mum!
Keith Braithwaite - Backing the Truth into a Corner
Backing the Truth into a Corner

Why do examples work so well?
And what can we do about it?
“I have nothing to sell. I am an entertainer. [...] I just want you to enjoy a point of view which I enjoy”
—Alan Watts
Cake or Biscuit?
Er...
Why?

- mainly because fresh cakes are soft and stale ones hard
- biscuits, vice-versa
Why?
- mainly because fresh cakes are soft and stale ones hard
- biscuits, vice-versa

Says who?
- Mr D. C. Potter, QC
  - United Biscuits v HM Commissioners of Revenue and Customs, Lon/91/160
Er...

Why?
- mainly because fresh cakes are soft and stale ones hard
- biscuits, vice-versa

Says who?
- Mr D. C. Potter, QC
  - United Biscuits v HM Commissioners of Revenue and Customs, Lon/91/160

Why would you care?
- Chocolate covered biscuits attract VAT
- Chocolate covered cakes do not
What Does This Tell Us?
What Does This Tell Us?

Categories are not out in the world waiting for us
He actually presents reasons why Jaffa cakes aren’t:

- too small
- eaten the wrong way
- little in common with other cakes
- not sold with other cakes
- etc.

And then concludes that they are cakes
- for the purposes of VAT
Concepts and Definitions

The “Classical” view of Concepts:

- Every category C has a Definition D
- D is a set of predicates
  \[ D = \{ p: \text{Predicate} \} \]
- An object o is understood as part of C iff o satisfies D
  \[ o \in C \Leftrightarrow \forall p \in D, p(o) \]
- Concepts are categorical
  - \( o \in C \lor o \notin C \) and nothing else can be said
The Convenience of Definitions

Technologists find definitions most agreeable:

- They are crisp, unambiguous
- They are compact and yet also universally quantified
- Predicates seem to lead very naturally to:
  - relational table signatures
  - class members
  - rules in a inference engine

<table>
<thead>
<tr>
<th>title_id</th>
<th>title</th>
<th>type</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>BU1032</td>
<td>The Busy Executive’s Database Guide</td>
<td>business</td>
<td>19.9</td>
</tr>
<tr>
<td>BU1111</td>
<td>Cooking with Computers</td>
<td>business</td>
<td>11.9</td>
</tr>
<tr>
<td>BU02075</td>
<td>You Can Combat Computer Stress!</td>
<td>business</td>
<td>2.9</td>
</tr>
<tr>
<td>BU7852</td>
<td>Straight Talk About Computers</td>
<td>business</td>
<td>19.9</td>
</tr>
<tr>
<td>MC2222</td>
<td>Silicon Valley Gastronomic Treats</td>
<td>med_cook</td>
<td>19.9</td>
</tr>
<tr>
<td>MC3021</td>
<td>The Gourmet Microwave</td>
<td>med_cook</td>
<td>2.9</td>
</tr>
<tr>
<td>MC3026</td>
<td>The Psychology of Computer Cooking</td>
<td>UNDECIDED</td>
<td></td>
</tr>
<tr>
<td>PC1035</td>
<td>But Is It User Friendly?</td>
<td>popular_comp</td>
<td>22.9</td>
</tr>
</tbody>
</table>
Definitions Force Users To Be “Precise”

Forces the user to think our way
- they have to accommodate our way of thinking

Use of definitions is a power play over our users
- are we afraid of the fuzziness of the world?
- are we afraid of our (in-)ability to deal with that?
An Alternative View

After 2300 years or so...

- Eleanor Rosch does experiments
- Just how do people understand the stuff of the world?

She made several discoveries:

1. categories are probabilistic
2. categories have structure
3. categories are fuzzy
4. categories relate to human capabilities
Rosch describes vertical and horizontal structure

- Some members of a category are better than others
Categories Have Structure

Rosch describes vertical and horizontal structure

- Some members of a category are better than others
Categories Have Structure

Rosch describes vertical and horizontal structure

- Some members of a category are better than others
Birds
Prototypical Birds
Categories Are Fuzzy

Near the edges, categories can blur into one another
Categories Relate to Human Capabilities

Remember the cakes?

Categories are grounded in human experience

- more than physics
Automated Tests Are...

User/acceptance/functional tests have escaped:
- no longer applied at the end of development
- no longer primarily about defects
Automated Tests are (Checked) Examples?

Checked examples metaphor emphasises:

- verification
- concreteness
- explicitness
Do We Have Any Clues Now About That?

If examples fit the way people think

- using them moves us towards our users
- connects us with their existential state
- corrects the power balance
Attack of the Clones

Mark Dalgarno
Software Acumen

Email: mark@software-acumen.com
Blog: blog.software-acumen.com
These types of clone

```c
int words, 1;
setbase %x, %y, %z; /* walking pointers to a, b, result */
words = a->size / SET_BITS;
if (a->size % SET_BITS)
  words++;

if (a->size ! b->size || b->size != result->size)
  panic("Size mismatch in %s: %s", _FILE_);

x = a->set;
y = b->set;
z = result->set;
```

```c
if (yyyn == YYFINAL)
  YYACCEPT;

/* Count tokens shifted since error/ after three, the */
/* status. */
if (yyerrstatus)
  yyerrstatus = YYERRSTATE;

/* Shift the look-ahead token. */
YY_SYMBOL_PRINT("Shifting", yytoken, &yyval, &yy); /* Discard the shifted token unless it is eol. */
if (yych == YYEOF)
  yych = YYEMPTY;
*yyvsp = yyval;
goto yynextstate;

/* yydefault -- do the default action for the current */
/* yydefault: */
  yyyn = yydefact[yynstate];
  if (yyyn == 0)
    goto yyreduce;
  goto yynreturn;
```

```c
if (yyyn == YYFINAL)
  YYACCEPT;

++yyvsp = yyval;

/* Shift the error token. */
YY_SYMBOL_PRINT("Shifting", yytoken, &yyval, &yy);

yystate = yyn;
goto yynnextstate;
```

```c
*.yyacceptlab = YYACCEPT comes here. */
*.yydefaultlab -- YYDEFAULT comes here. */
*.yyabortlab -- YYABORT comes here. */
*yyexhaustedlab -- memory exhaustion comes here. */
```
What’s the problem?

- Cloning code increases your code size, this can make it
  - harder to understand,
  - slower to build &
  - have a larger footprint (expensive in an embedded setting).

- Cloning code increases your maintenance effort
  - Having to fix bugs in more than once place
  - Having to find bugs in more than once place
  - Having to add features in more than one place
How big is the problem?

“I am a clone I’m not alone. Is that the spirit of the age?”

Robert Calvert (1945-1988), (Hawkwind) *Spirit of the Age* from the album *Quark, Strangeness & Charm* (1977 Charisma)
Our recent industrial experience

11 C/C++ projects varying from 10 KLOC to 800 KLOC
The Copy-and-Paste Programming Manifesto

I will not copy and paste code

See http://tinyurl.com/d5cd26 for further resources or email mark@software-acumen.com
Guy Bolton-King - Test Your Java With Java
Test your Java code with Scala

Internal DSLs are easy in Scala
class MustBe(val name: String) {
  def mustEqual(s: String) {
    if (s != name)
      throw new RuntimeException
        (s + " was not equal to " + name)
  }
}

val m = new MustBe("foo")

m.mustEqual("foo")

m mustEqual "foo"

implicit def convertStringToMustBe(s: String) =
  new MustBe(s)
public class Stack<T>
{
    private ArrayList<T> _stack = new ArrayList<T>();

    static class Empty extends RuntimeException {}

    public void push(T x) { _stack.add(x); }

    public T pop() {
        if (_stack.size() == 0)
            throw new Empty();
        return _stack.get(_stack.size() - 1);
    }

    public int size() { return _stack.size(); }

    public boolean empty() { return size() == 0; }
}
public class StackTestJUnit {
    @Test
    public void shouldBeEmptyAfterCreation() {
        Stack<Integer> stack = new Stack<Integer>();
        assertTrue(stack.size() == 0);
    }

    @Test
    public void shouldIncreaseInSizeBy1AfterAPush() {
        Stack<Integer> stack = new Stack<Integer>();
        stack.push(1);
        assertTrue(stack.size() == 1);
    }

    @Test
    public void shouldDecreaseInSizeBy1AfterAPop() {
        Stack<Integer> stack = new Stack<Integer>();
        stack.push(1);
        stack.pop();
        assertTrue(stack.empty());
    }
}

class StackTest extends Suite with Spec with ShouldMatchers {

  describe("a stack") {
    it("should be empty after creation") {
      val stack = new Stack[Int]
      stack should have size 0
    }

    it("should increase in size by 1 after a push") {
      val stack = new Stack[Int]
      stack.push(1)
      stack should have size 1
    }

    it("should decrease in size by 1 after a pop") {
      val stack = new Stack[Int]
      stack.push(1)
      stack.pop()
      stack should be ('empty)
    }
  }
}
Tests Run: 3  Expected: 3  Failed: 1

Reports:
- Run Starting
- StackTest:
  - a stack
    - should be empty after creation
    - should increase in size by 1 after a push
- StackTest: a stack should decrease in size by 1 after a pop
- Run Completed

Details:
- Report: Test Failed
- Name: StackTest: a stack should decrease in size by 1 after a pop
- Message: com.waftex.rubbish.Stack@19d6af was not empty
- Line: (StackTest.scala:24)
- Date: Fri Apr 24 16:15:46 BST 2009
- Thread: Thread-1
- Exception: org.scalatest.TestFailedException
object should have length (3)

string should startWith ("Hello")

string should fullyMatch regex (\(-\)?(\d+)(\d*)?\)

one should be < (7)

emptySet should be ('empty)

seven should be (6 plusOrMinus 2)

map should contain key (1)

map should contain value ("Howdy")
object Lunar extends Baysick {
    def main(args:Array[String]) = {
        10 PRINT "Welcome to Baysick Lunar Lander v0.9"
        20 LET ('dist := 100)
        30 LET ('v := 1)
        40 LET ('fuel := 1000)
        50 LET ('mass := 1000)
        
        60 PRINT "You are drifting towards the moon."
        70 PRINT "You must decide how much fuel to burn."
        80 PRINT "To accelerate enter a positive number"
        90 PRINT "To decelerate a negative"
        
        100 PRINT "Distance " % 'dist % " km, " % "Velocity " % 'v % " km/s, " % "Fuel " % 'fuel
        110 INPUT 'burn
        120 IF ABS('burn) <= 'fuel THEN 150
        130 PRINT "You don't have that much fuel"
        140 GOTO 100
        150 LET ('v := 'v + 'burn * 10 / ('fuel + 'mass))
        160 LET ('fuel := 'fuel - ABS('burn))
        170 LET ('dist := 'dist - 'v)
        180 IF 'dist > 0 THEN 100
        190 PRINT "You have hit the surface"
        200 IF 'v < 3 THEN 240
        210 PRINT "Hit surface too fast (" % 'v % ") km/s"
        220 PRINT "You Crashed!"
        230 GOTO 250
        240 PRINT "Well done"
        
        250 END
    }
}

RUN