Software Engineering at AxaRosenberg

Mark Howard
Director of Software Engineering
Barr Rosenberg Research Center
The Setting

- AxaRosenberg is an institutional investment equity management firm which manages over $120 Billion with offices in California, London, Tokyo, Hong Kong, Singapore and Sydney. Since the inception of the company in 1985 the goal has been to have the company driven by an expert system.
The Turing Test

• In 1950 Alan Turing proposed a test for artificial intelligence, called the imitation game. A human being and a computer would be interrogated under conditions where the interrogator would not know which was which, the communication being entirely by messages and queries. Turing argued that if the interrogator could not tell which was which then it would be reasonable to call the machine intelligent.
Automation

• Our company is made up of many participants. The participants interact with each other through messages and queries. Some of these participants are humans in a particular role, and others are computer programs. In theory automating a human role could be accomplished by replacing it with a program and verifying that it passes the Turing test.
Reinventing Yourself

When we were young and innocent we thought that the main issue was

• Process
  – How do we automate what we do?

Now that we are older and wiser we realize that the real issue is

• Knowledge
  – How do we represent what we know?
Knowledge vs Process

• Knowledge is cumulative
  – Process is transitory
• Knowledge is global
  – Process is local
• Knowledge is necessary for process
  – Process is not necessary for knowledge
• Knowledge inspires process
  – Process often obfuscates knowledge
Knowledge Base

- So of course our only choice was to write our system in Eiffel
  - No stored procedures or triggers (do not put business logic in a database)
  - No third party applications
What we expect from knowledge

• Fidelity
  – Say what you mean

• Integrity
  – Mean what you say

• Coherence
  – Changes should preserve fidelity and integrity
The goal: Do something to each filename in a directory
Performing dir on the directory yields
- 03/01/2004 03:30a 3,964 $AXA022720040301.CSV
- 04/01/2004 03:00a 3,970 $AXA033120040401.CSV
- 05/03/2004 03:00a 3,970 $AXA04302004040503.CSV
- 06/01/2004 04:00a 3,970 $AXA053120040601.CSV
- 04/02/2004 03:00a 3,970 $AXA20040401.CSV
A VB script does the following
Fidelity
Say what you mean

• Months later the program breaks and we discover that we are trying to do something with the filename 3,964. After much searching we discover that “do something with each filename” was implemented by the VB script as “do something to each 4th column”
Fidelity
Say what you mean

• Notice that the program still does exactly what it says it does: “Do something with each 4th column of the result of dir”, but not what the programmer meant, which was “Do something with each filename in the result of dir”
Fidelity
Say what you mean

• Dir now yields
  • 03/01/2004 03:30 AM 3,964 $AXA022720040301.CSV
  • 04/01/2004 03:00 AM 3,970 $AXA033120040401.CSV
  • 05/03/2004 03:00 AM 3,970 $AXA043020040503.CSV
  • 06/01/2004 04:00 AM 3,970 $AXA053120040601.CSV
  • 04/02/2004 03:00 AM 3,970 $AXA20040401.CSV

• Instead of
  • 03/01/2004 03:30a 3,964 $AXA022720040301.CSV
  • 04/01/2004 03:00a 3,970 $AXA033120040401.CSV
  • 05/03/2004 03:00a 3,970 $AXA043020040503.CSV
  • 06/01/2004 04:00a 3,970 $AXA053120040601.CSV
  • 04/02/2004 03:00a 3,970 $AXA20040401.CSV
Integrity
Mean what you say

```c
void swap (int *x, int * y)
{
    int z;
    z = *x;
    *x = *y
    *y = z
}
```
The Gossip Game

• The following is an English sentence which was first translated to Spanish by an online translator, and then back to English by the same application
  – our asideros enlarged by 300 actions after the party

• The original:
  – our holdings were increased by 300 shares after the split
There used to be a satirical political party in Canada known as the Rhino party. One year their platform was to

- “Bring back the British convention of driving on the left side of the road. To minimize the impact trucks and buses would be phased in the second year.”
Fidelity
Say what you mean

• Conventional wisdom in software design dictates thin interfaces
  – Usual argument is that you do not want changes in one part of your system to break other parts of your system

• On the other hand it is essential that the different parts of our system are talking about the same things
Fidelity
Say what you mean

• Thin interfaces throw away information by design. How do you ensure that no information is lost without building a monolithic system?
• Use the compiler to ensure that you are talking about the same objects.
Fidelity
Say what you mean

• Our code base always satisfies two constraints
  – Integration always compiles, in other words the project with root NONE and the entire system as a cluster
  – All descendents of RP_TEST_CASE always succeed

• It doesn’t matter whether you are two objects in the same executable, or two processes passing objects via storables … you know that you are talking about the
Integrity
Mean what you say

• In order to mean what we say, we need to agree on what we mean. This requires collaboration
  – It is easier to recognize a good idea than it is to come up with one
  – It is easier to see what is wrong with other people’s ideas than your own
• But…Functionality is a product of shared knowledge, not shared fantasy
Integrity
Mean what you say

• A few years ago we started using Eiffelstudio as a tool for development teams. The team would consist of programmers and business experts. They would convene in a room with a projector and a keyboard. The project being worked on would be brought up in Eiffelstudio and projected on the screen, and there would be one person driving the session
Integrity
Mean what you say

• We discovered very quickly that flowcharts, uml, bubble charts, visio etc are unnecessary at best, usually misleading and always time consuming. We use design by contract to develop the classes, and stub out features which require expertise to implement
Integrity
Mean what you say

• The compiler fills the same role as a theorem prover … you don’t trust anything until the compiler verifies it. The project is compiled continuously.

• This was very frustrating at first, since the compiler exposes differences of opinion that you do not see otherwise… it is easy to gloss over subtle distinctions verbally
Integrity
Mean what you say

- Between sessions experts would implement the stubbed out features
- This method of building expert systems had as a side effect that we were building system experts
Integrity
Mean what you say

• We created a class RP_UNFINISHED (later implemented by ISE as REFACTORING_HELPER) with a feature
  – to_be_implemented(a_string : STRING)
  – Each developer has their own class which inherits RP_UNFINISHED
  – RP_UNFINISHED has a feature for each developer
  – If you inherit RP_UNFINISHED then you can call
    • mark_howard.to_be_implemented(“fix this code”)
Integrity
Mean what you say

• Squabbling is good
  – Nobody knows everything
  – Obvious truths are often false
  – The compiler is the final arbiter
There is an island with the following properties

- Gossip is extremely effective: If any man cheats on his wife, every woman other than his wife knows about it
- Punishment is absolute: If a woman discovers that her husband has been cheating on her then she must shoot him at midnight

- The island survives in perfect harmony for hundreds of years until a preacher comes and gives a sermon … he admonishes them for their sinfulness and states that he knows for a fact that at least one man on the island is unfaithful

- Nothing happens for seven days, but on the seventh day shots ring out at midnight

How many husbands were shot and why?
Coherence
Evolve gracefully

- This illustrates two things
  - Change often causes trouble
  - It is not always clear what change caused the trouble
Coherence
Evolve gracefully

• We ensure coherence by
  – All developers check out only those classes they are working on, and put them in an override cluster
  – Before checking classes in they are required to integrate against all classes in our system. This is accomplished by having a project with root class NONE, root cluster our repository and override cluster with the classes being checked in
Coherence
Evolve gracefully

• We have built a tool, rose studio, for facilitating the process
  – We can integrate against only the classes being checked in, in other words we can specify a virtual override consisting of only those classes in the override we want to check in right now
  – We can invoke all descendents of RP_TEST_CASE in our system which are in the transitive closure of clients of the classes being checked in
Coherence
Evolve gracefully

• Developers are impacted by any changes as soon as they happen, so they do not spend weeks developing code which they will only find out is incompatible with other changes when they go to merge … first one in wins
Coherence
Evolve gracefully

• By requiring that you integrate, if you make changes which affect clients of your class then it is your responsibility to make sure that they continue to work. Note that if you change the signature of a feature then you are the most likely person to understand how the old call needs to be translated to the new call.
Migration

• A surgeon was once berated by an auto mechanic who claimed that they were paid vastly different amounts for doing essentially the same thing. The surgeon responded “try rebuilding an engine while it is running”
Migration

• We have been in the process of migrating and enhancing a complex system from VMS/FORTRAN to platform-independent Eiffel for several years
Migration

- 2 million lines of FORTRAN
- 1 million lines of C
- Hundreds of Rdb databases
- Global sections
- Rms index files
Migration

• Using Eiffel we have been able to transform the existing system rather than replace it
  – Use Cecil to call Eiffel on VMS, using VMS Eiffel
  – Use general store and eiffelnet to pass objects between VMS and windows
  – Use callouts to C on windows to talk to Rdb directly
System integrity
Code repository integrity
Source code integrity
Application integrity
Process integrity
Execution integrity
Data integrity
Extensively used features

- Assertions
- Design by contract
- Multiple inheritance
- Genericity and generic creation
- Agents
- Storables
  - MISMATCH_CORRECTOR, but use ONLY for conversion
Extensively used features

- Override clusters
- Anchored types
- Once features
- Deferred features
- Labeled tuples
- External calls
- Platform independence