Evolving into Embedded Development

Matt Fletcher
Atomic Object
Agile 2007
About us

- Atomic Object develops custom software
- Savant Automation produces automated-guided vehicles

- Savant Automation produces automated-guided vehicles
Speed control

- Accepts speed requests from main computer
- Translates speed into voltages the drives understand
- Input and output:
  - Digital and analog
  - CAN bus
- Microchip PIC18F4480
  - 768 bytes RAM
  - 16K bytes flash
Speed control
Tools

- Microchip C compiler and simulator
- Ruby
- Rake - Ruby make
  - Main build script
  - Easy injection of Ruby scripts into the build process
- Other Ruby scripts
Unit testing

- Created our own unit testing macros and runner
- Each test executable ran through the GUI simulator
- Blend of state-based and interaction-based testing
- Model-Conductor-Hardware made testing easier
- All of the code was developed test-first
Model-Conductor-Hardware

composed into

Conductor

Model

Hardware
Argent

- Inline code generation
- Ruby script
- Fast; easy to use
```c
void testVoltageCalculatorWithPositiveVoltage()
{
    ...
}

void testVoltageCalculatorWithNegativeVoltage()
{
    ...
}
```

```c
//[[[$argent require "tests.rb"; generate_tests;$]]
int main(int argc, char* argv[]){
    RUN_TEST(testVoltageCalculatorWithPositiveVoltage);
    RUN_TEST(testVoltageCalculatorWithNegativeVoltage);
}
//[[[$end$]]]```
Positives

- Test-driven development resulted in good code
- Rake made for simple, straightforward builds
- Argent automatically detected and executed tests
- We learned a lot about programming this PIC
Negatives

- State-based tests were hard to maintain
- Manual system testing
- GUI simulator
  - slow
  - brittle
Battery monitor
Battery monitor

Analog input
Battery monitor

Analog input

Digital output
Battery monitor

- Analog input
- Digital output
- CAN output
Battery monitor

Analog input
Digital output
CAN output

Microchip PIC18F4480
New tools

- IAR Systems workbench for PIC
  - better compiler
  - command-line simulator
- New simulator made the tests run much faster
System testing

Create system test for new feature

Run the system test

Pass?

Yes

Develop supporting units

No

Pass?
Supporting system tests

- **Software:**
  - Domain-specific language written using Ruby

- **Hardware**
  - Digital and analog input and output
  - CAN input and output
  - PIC programmer
pCAN-USB
1. Program newest firmware
2. Output known voltages
3. Read battery level
4. Verify battery level matches expected

System test running on host PC

Running firmware
System test-driven development

- Test automation
- Choosing support hardware with command line and programmatic interfaces
- Diligence. System-test first!
Unit testing

• Much more interaction-based testing than speed control

• Mocks created automatically
  - Ruby script scans header file
  - Creates mock header and source

• Naming conventions dictate how to declare functions and how to call mock functions

• Argent generated code to automatically verify mocks
Continuous integration

<table>
<thead>
<tr>
<th>Al n Bobs</th>
<th>aLive 3</th>
<th>AO website</th>
<th>ARS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARS Peer Review</td>
<td>batterymonitor</td>
<td>BLV Parser Builder</td>
<td>BM Demo Agent Linux</td>
</tr>
<tr>
<td>BM Demo Agent Solaris</td>
<td>BM Demo Agent Windows</td>
<td>CraneLogic</td>
<td>DCI monitor</td>
</tr>
<tr>
<td>DCI server</td>
<td>DIY</td>
<td>EFD Builder linux</td>
<td>EFD Builder rel2 linux</td>
</tr>
<tr>
<td>EFD Builder windows</td>
<td>esc07_demo</td>
<td>Fantasy Real Estate</td>
<td>Fantasy Real Estate Data</td>
</tr>
<tr>
<td>Fishladder aLive2</td>
<td>GLSEC</td>
<td>Hardmock</td>
<td>HNRC</td>
</tr>
<tr>
<td>Holistic</td>
<td>Joe Cartoon</td>
<td>minilab</td>
<td>MMPC</td>
</tr>
<tr>
<td>Notions</td>
<td>Notions DVD Updates</td>
<td>Notions Monitor App</td>
<td>pcan</td>
</tr>
<tr>
<td>Project Vision</td>
<td>Punchit 3</td>
<td>Puritan</td>
<td>spiderweb</td>
</tr>
<tr>
<td>stoplightd</td>
<td>SYSTIR</td>
<td>The Common</td>
<td>unity_iar</td>
</tr>
<tr>
<td>urc</td>
<td>VCL site</td>
<td>X12 parser</td>
<td>XPWM</td>
</tr>
</tbody>
</table>
Positives

• System tests provided automated regression testing of complete features
• Interaction-based testing made tests easier to write and less brittle
• Command line simulator made tests run faster
• Cool Ruby extensions
Negatives

- Expensive compiler and simulator (but worth every penny)
- Small project
  - did not last very long
  - did not stress our system testing strategy
Build your own tools

• Solve specific problems with custom tools
• Automate tedious processes
• Generate repetitive code
Resources

• Rake
  - rake.rubyforge.org

• Argent
  - rubyforge.org/project/argent

• Minilab Ruby gem
  - minilab.rubyforge.org
Resources

- Atomic Object embedded
  - www.atomicobject.com/pages/Embedded+Software
  - papers
  - Embedded Systems Conference 2007 demo project

- Agile Embedded Yahoo! group:
  - tech.groups.yahoo.com/group/AgileEmbedded
Thanks

- Bill Bereza, Mike Karlesky, and Greg Williams

- Matt Werner and Andrew Black

- Chad Fowler and Carl Erickson
Questions?