Applying Aspects in a Legacy Environment

A Reengineering Case Study

Bram Adams, Kris De Schutter and Andy Zaidman

13 december 2005, VUB

Architectural Resources for the Restructuring and Integration of Business Applications
Overview

The case
The groundwork
The results
Conclusion
Architectural Resources for the Restructuring and Integration of Business Applications
Kava Application -- Background

- ICA project
  - Migration to GCC (ANSI-C)
  - 407 C Modules (determined statically)
  - 269 Makefiles (determined statically)

- TDFS
  - 15 participating modules
  - 237 participating procedures
The Case: TDFS

- Produces a digital and detailed invoice of all prescriptions for the healthcare insurance institutions.
- Often used as a final check to see whether adaptations in the system have any unforeseen consequences.
- Should be considered as a functional application, but also as a form of regression test.
Task Description

Apply dynamic analyses:

- Frequency analysis
  - GOAL: determine cohesiveness of structure

- Webmining coupling metrics
  - GOAL: find most important modules in system

- Coverage analysis
  - GOAL: determine test coverage (optional: detect dead code)
The Groundwork

Architectural Resources for the Restructuring and Integration of Business Applications
The Groundwork

Goal: a trace of the application.

Approach: use a simple **tracing aspect**.
The Tracing Aspect

```
ReturnType around tracing_rest (ReturnType)
on (Jp):

  call(Jp,"^(!.*printf$|.*scanf$).*$")
  && type(Jp,ReturnType)
  && !str_matches("void",ReturnType)

  {
    FILE* fp=fopen("/kava/home/uial/log.txt","a");
    ReturnType i;
    fprintf (fp, "before ( %s in %s ) \n",
               Jp->functionName,Jp->fileName);
    fflush(fp);
    i = proceed ();
    fprintf (fp,"after ( %s in %s ) \n",
               Jp->functionName,Jp->fileName);
    fclose(fp);
    return i;
  }
```
Aspectizing The Build Process

```
makefile
GCC  ESQL  ...
.O    a.out  ...
```

```
Aspicere
GCC  ESQL  ...
.O    a.out  ...
```
Build Process Integration (1)

```bash
gcc -c -o file.o file.c

gcc -E -o tempfile.c file.ccp tempfile.c file.c
aspicere -i file.c -o file.c \
  -aspects aspects.lst
```
```bash
gcc -c -o file.o file.c
```
Difficulties

- **Makefile adaptation:**
  - Full automation requires on-site configurability
  - Vendor-specific tools (ESQL, ...), see next slide
  - Our weaving framework crosscuts makefile hierarchy
  - "inline" weaving

- **Build time:** **15min → 17h38m**
  - Matching utterly slow
  - Weaver is preprocessor

- **Complexity:**
  - Mixture of ANSI and traditional C
  - Complex type inference rules
Build Process Integration (2)

```bash
.ec.o:
    $(ESQL) -c $*.ec
    rm -f $*.c

.ec.o:
    $(ESQL) -e $*.ec
    chmod 777 *
    cp `ectoc.sh $*.ec` $*.ecesql -nup $*.ec $(C_INCLUDE)
    chmod 777 *
    cp `ectoicp.sh $*.ec` $*.ec
    aspicere -verbose -i $*.ec -o \
    `ectoc.sh $*.ec` -aspects aspects.lst
    gcc -c `ectoc.sh $*.ec`
    rm -f $*.c
```
It Worked!

± 486 000 000 procedure calls → 90 GB

= 972 000 000 events
The Results

Architectural Resources for the Restructuring and Integration of Business Applications
Frequency Analysis

-created “frequency clusters”
-All methods executed the same number of times are in 1 cluster
Frequency Clusters

- 237 procedures of which 160 were grouped in 1 of 25 frequency clusters (i.e. 67.5% catalogued)

Categories

- 2 weakly coupled (≤ 50% of procedures in same module)
- 10 normally coupled (> 50%)
- 13 strongly coupled (100%) ➔ cohesive!
Lessons Learned from FSA

- Related procedures can easily be found
- Kava application is well-built, as witnessed by the high number of cohesive clusters
- 4 clusters contain a lot of procedures
  - Possibly a maintenance problem
  - Does point to good decomposition
## Webmining

<table>
<thead>
<tr>
<th>Module</th>
<th>Aut</th>
<th>Hub</th>
</tr>
</thead>
<tbody>
<tr>
<td>/ica/project/deelproject/TDFS/sources/e_tdfs_mut1.c</td>
<td>0.915478</td>
<td>0.814941</td>
</tr>
<tr>
<td>/ica/project/deelproject/TDFS/sources/tdfs_mut1_form.c</td>
<td>0.872067</td>
<td>0.45397</td>
</tr>
<tr>
<td>/ica/project/deelproject/TDFS/sources/tdfs_bord.c</td>
<td>0.198554</td>
<td>0.397726</td>
</tr>
<tr>
<td>/ica/project/deelproject/TDFS/sources/tdfs_mut2.c</td>
<td>0.594401</td>
<td>0.164278</td>
</tr>
<tr>
<td>/ica/project/algemeen/Show_listing_box/tools.c</td>
<td>0.198554</td>
<td>0.164278</td>
</tr>
<tr>
<td>/ica/project/deelproject/batch/PROCESSOR/RELEASE/io.c</td>
<td>0.716924</td>
<td>0.12548</td>
</tr>
<tr>
<td>/ica/schermen/cprogs/csrout.c</td>
<td>0.198554</td>
<td>0.0321257</td>
</tr>
<tr>
<td>/ica/project/algemeen/apoteek/tarpargeg/tarpargeg.c</td>
<td>0.55099</td>
<td>0</td>
</tr>
<tr>
<td>/ica/schermen/cprogs/cs routines.c</td>
<td>0.213674</td>
<td>0</td>
</tr>
<tr>
<td>/ica/project/algemeen/strcpy/UW_strncpy.c</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>/ica/project/algemeen/Show_listing_box/td.ec</td>
<td>0.198554</td>
<td>0</td>
</tr>
<tr>
<td>/ica/project/deelproject/batch/PROCESSOR/RELEASE/cache.c</td>
<td>0.716924</td>
<td>0</td>
</tr>
<tr>
<td>/ica/project/algemeen/decfties/decfties.c</td>
<td>0.766083</td>
<td>0</td>
</tr>
<tr>
<td>/ica/project/deelproject/tapes/sources/weglf.c</td>
<td>0.198554</td>
<td>0</td>
</tr>
<tr>
<td>/ica/schermen/cprogs/get_request.c</td>
<td>0.198554</td>
<td>0</td>
</tr>
</tbody>
</table>
Feedback on Webmining

We interviewed two Kava developers before showing them our result set. Question: which module(s) is(are) most important?

- D1 mentioned `e_tdfs_mut1.c` and `tdfs_mut2.c`
- D2 mentioned `e_tdfs_mut1.c`

Our result set ranks them in the top 4, with `e_tdfs_mut1.c` as the most important.

Counter indication: filenames contain “tdfs”.
## Coverage Analysis

<table>
<thead>
<tr>
<th>Average</th>
<th>Median</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Procedure coverage (%)</th>
<th>Statement coverage (%)</th>
</tr>
</thead>
</table>

Sorry… non disclosure agreement 😞
## Coverage Analysis

### Code Coverage Calculator

<table>
<thead>
<tr>
<th>Function</th>
<th>Coverage</th>
<th>Statement Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>File1/General</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetInto</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MarkInit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OpenMuxi</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write10Rec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write20, 40, 80Rec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write80Rec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NegativeCloseSetTab</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CreateDest</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CloseRemoveMntini</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OpenMnt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CloseRemoveMntini</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ReadBlockErr</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ReadBlock</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetFile</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RemoveUnknown</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RemoveNewRec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EndObj</td>
<td></td>
<td></td>
</tr>
<tr>
<td>System</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Precheck         modules</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Results of the Coverage Analysis

- TDFS is considered as test scenario
- At first sight, coverage is disappointing
- However, after *Belgian Franc to Euro* conversion, a lot of dead code remained
  - filenames prepended with *e_* point to new versions for Euro
  - old versions are not covered, i.e. tested
Conclusion

Architectural Resources for the Restructuring and Integration of Business Applications
Conclusion

- The webmining approach results in a ranking of modules according to their importance from a program comprehension point of view.
- The frequency analysis approach allowed to easily audit the system's internal structure.
- Test coverage measurements managed to uncover some dead code.
- Our AOP framework allowed us to do a clean and non-intrusive trace of the entire application.
- But, …
# Effort Analysis

<table>
<thead>
<tr>
<th></th>
<th>normal</th>
<th>with AOP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build cycle</td>
<td>15 → 20 min</td>
<td>17h38m</td>
</tr>
<tr>
<td>Execute scenario</td>
<td>1h30m</td>
<td>+- 7h</td>
</tr>
<tr>
<td>Logfile size</td>
<td>90 GB (600 MB compressed)</td>
<td></td>
</tr>
<tr>
<td>Code coverage</td>
<td>5h</td>
<td></td>
</tr>
<tr>
<td>Frequency analysis</td>
<td>5h</td>
<td></td>
</tr>
<tr>
<td>Webmining analysis</td>
<td>10h</td>
<td></td>
</tr>
</tbody>
</table>
Submitted Papers

- *AOP for Legacy Environments, a Case Study, in European Interactive Workshop on Aspects in Software, EIWAS '05* (Brussels).


- *Regaining Lost Knowledge through Dynamic Analysis and Aspect Orientation: An Industrial Experience Report, CSMR '06* (Bari, It.).
Q&A

Architectural Resources for the Restructuring and Integration of Business Applications