Arriba

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

Arriba

The Case

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)

Arriba

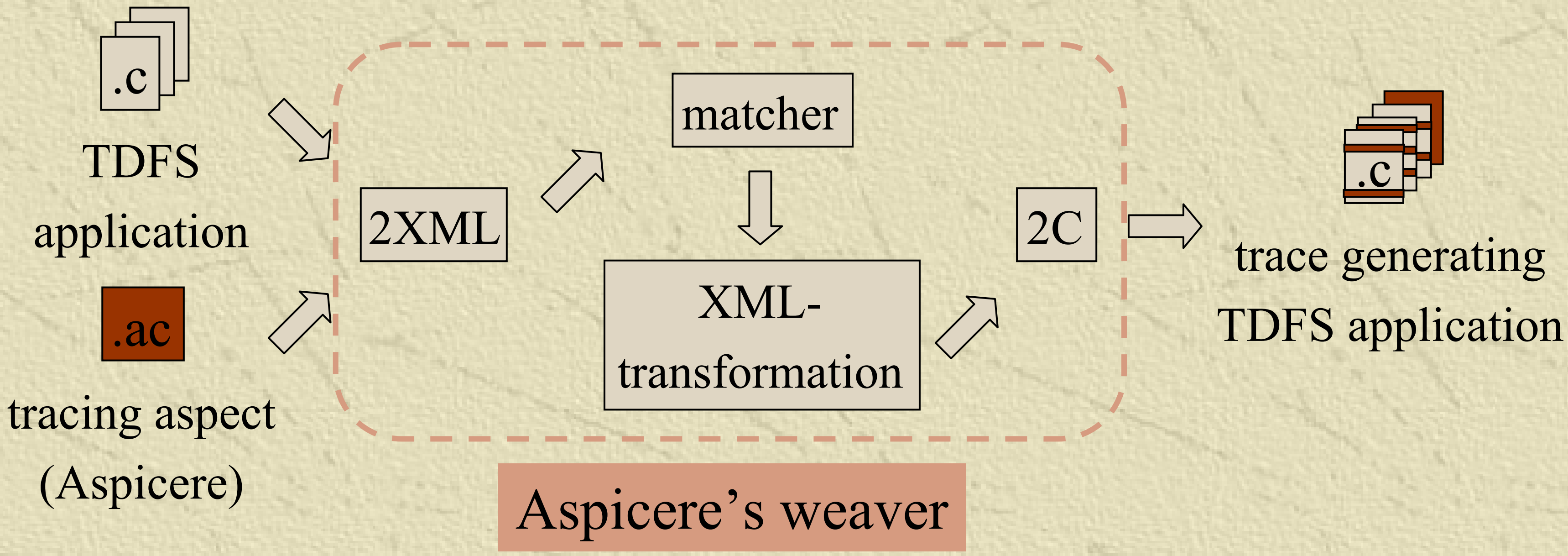
The Groundwork

Architectural Resources for the Restructuring and Integration of Business Applications

Arriba

The Groundwork

- Goal: a trace of the application.
- Approach: use a simple **tracing aspect**.



The Tracing Aspect

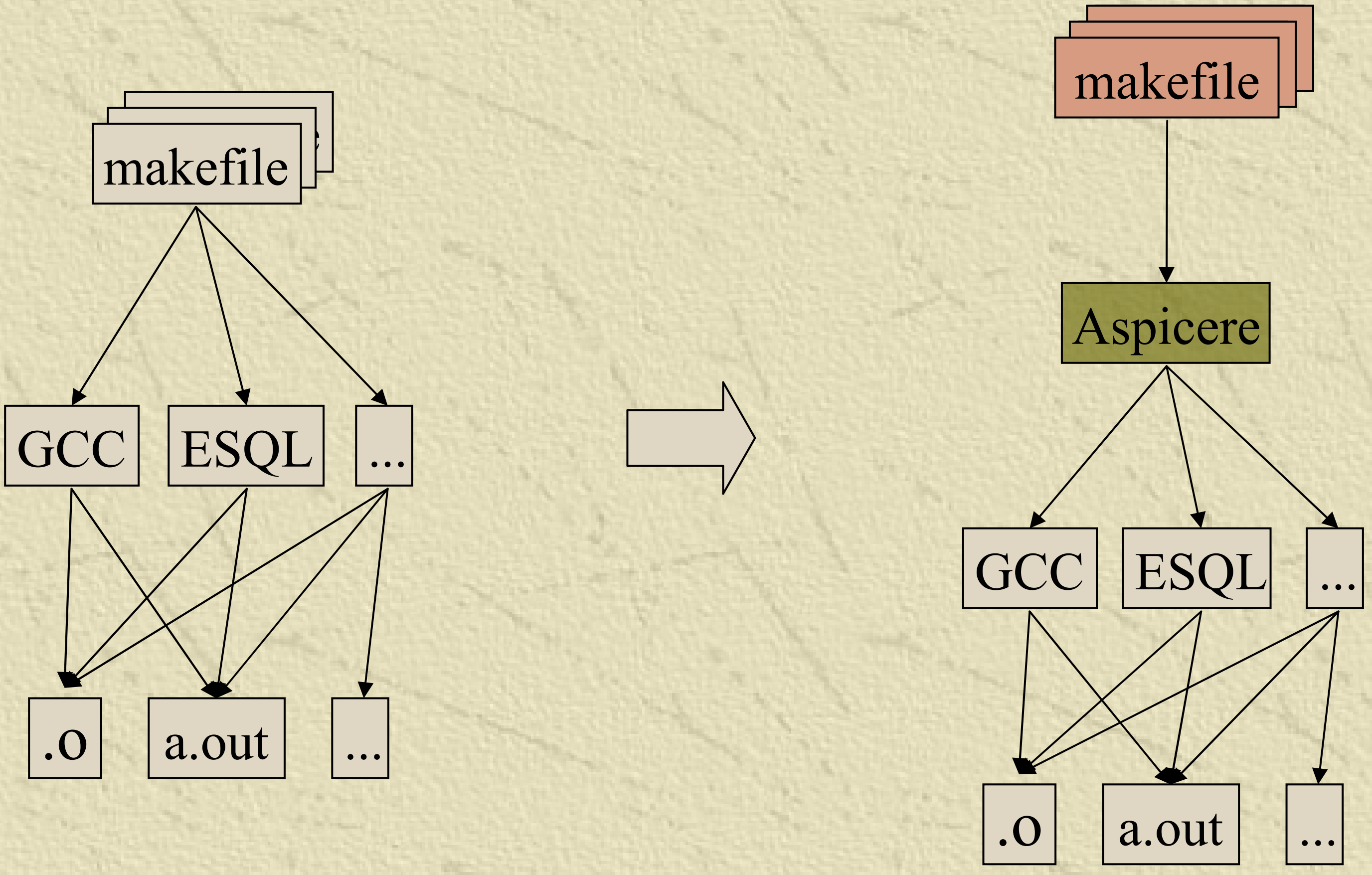
```
ReturnType around tracing_rest (ReturnType)
on (Jp):
  call(Jp,"^(?!.*printf$|.*scanf$).*$")
  && type(Jp,ReturnType)
  && !str_matches("void",ReturnType)
{
  FILE* fp=fopen("/kava/home/uia1/log.txt","a");
  ReturnType i;
  fprintf (fp, "before ( %s in %s ) \n",
             Jp->functionName,Jp->fileName);
  fflush(fp);
  i = proceed ();
  fprintf (fp,"after ( %s in %s ) \n",
             Jp->functionName,Jp->fileName);
  fclose(fp);
  return i;
}
```

advice signature

pointcut

advice body

Aspectizing The Build Process



Build Process Integration (1)

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

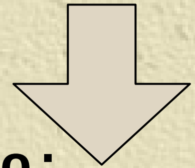
Difficulties

- 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)

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



```
.ec.o:
  $(ESQL) -e $*.ec
  chmod 777 *
  cp `ectoc.sh $*.ec` $*.ec
  esql -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!

```
before ( fgets in /ica/project/deelproject/TDFS/sources/tdfs_mut2.c )
after ( fgets in /ica/project/deelproject/TDFS/sources/tdfs_mut2.c )
before ( RemoveNewline in /ica/project/deelproject/TDFS/sources/tdfs_mut2.c )
after ( RemoveNewline in /ica/project/deelproject/TDFS/sources/tdfs_mut2.c )
before ( UW_atoi in /ica/project/deelproject/TDFS/sources/tdfs_mut2.c )
  before ( atoi in /ica/project/algemeen/strcpy/UW_strncpy.c )
  after ( atoi in /ica/project/algemeen/strcpy/UW_strncpy.c )
after ( UW_atoi in /ica/project/deelproject/TDFS/sources/tdfs_mut2.c )
before ( UW_atoi in /ica/project/deelproject/TDFS/sources/tdfs_mut2.c )
  before ( atoi in /ica/project/algemeen/strcpy/UW_strncpy.c )
  after ( atoi in /ica/project/algemeen/strcpy/UW_strncpy.c )
```

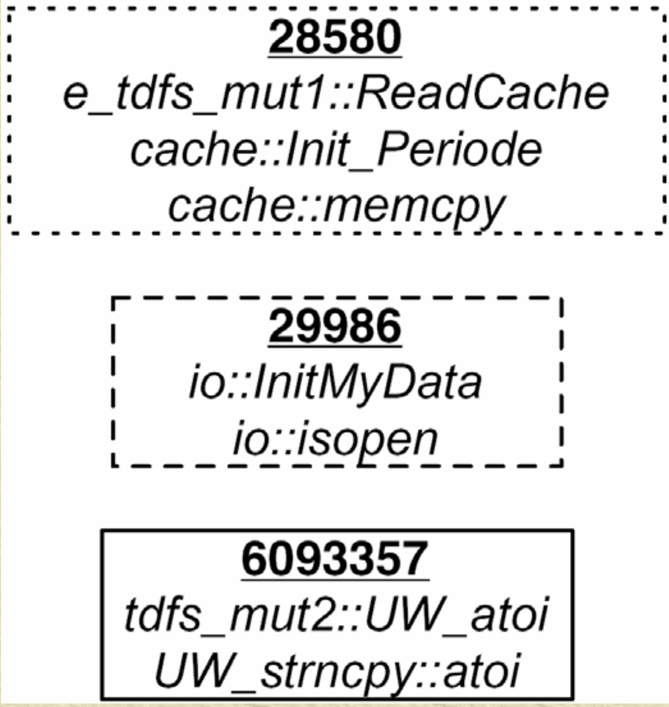
± 486 000 000 procedure calls → 90 GB

= 972 000 000 events

The Results

Frequency Analysis

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



28580
e_tdfs_mut1::ReadCache
cache::Init_Periode
cache::memcpy

29986
io::InitMyData
io::isopen

6093357
tdfs_mut2::UW_atoi
UW_strncpy::atoi

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

Module	Aut	**Hub**
/ica/project/deelproject/TDFS/sources/e_tdfs_mut1.c	0.915478	**0.814941**
/ica/project/deelproject/TDFS/sources/tdfs_mut1_form.c	0.872067	**0.45397**
/ica/project/deelproject/TDFS/sources/tdfs_bord.c	0.198554	**0.397726**
/ica/project/deelproject/TDFS/sources/tdfs_mut2.c	0.594401	**0.164278**
/ica/project/algemeen/Show_listing_box/tools.c	0.198554	**0.164278**
/ica/project/deelproject/batch/PROCESSOR/RELEASE/io.c	0.716924	**0.12548**
/ica/schermen/cprogs/csrout.c	0.198554	**0.0321257**
/ica/project/algemeen/apoteek/tarpargeg/tarpargeg.c	0.55099	**0**
/ica/schermen/cprogs/csroutines.c	0.213674	**0**
/ica/project/algemeen/strcpy/UW_strncpy.c	1	**0**
/ica/project/algemeen/Show_listing_box/td.ec	0.198554	**0**
/ica/project/deelproject/batch/PROCESSOR/RELEASE/cache.c	0.716924	**0**
/ica/project/algemeen/decfties/decfties.c	0.766083	**0**
/ica/project/deelproject/tapes/sources/weglf.c	0.198554	**0**
/ica/schermen/cprogs/get_request.c	0.198554	**0**

Feedback on Webmining

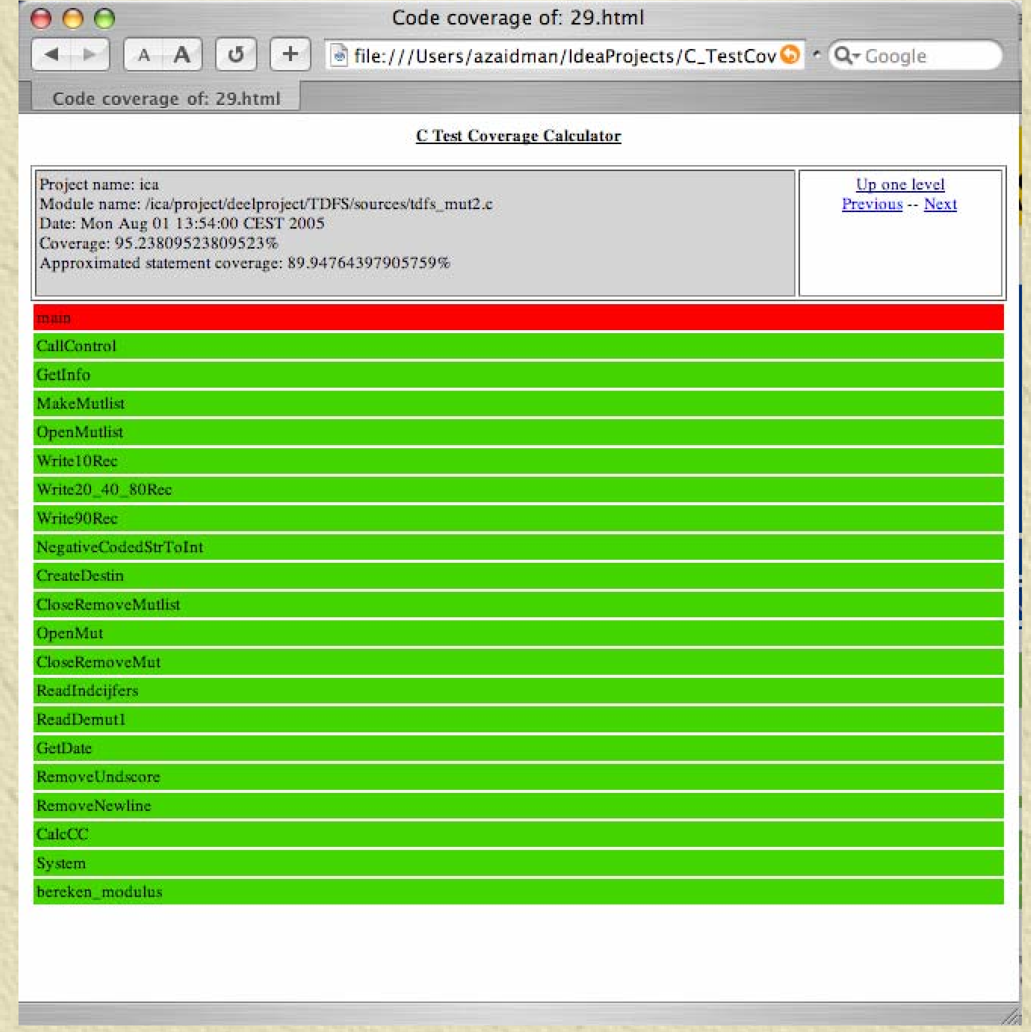
- We interviewed two Kava developers before showing them our result set. Question: which module(s) is(/are) most important?
  - D1 mentioned *e_tdfs_mut1.c* and *tdfs_mut2.c*
  - D2 mentioned *e_tdfs_mut1.c*
- Our result set ranks them in the top 4, with *e_tdfs_mut1.c* as the most important.
- Counter indication: filenames contain “tdfs”.

Coverage Analysis

	Procedure coverage (%)	Statement coverage (%)
Average		
Median		

Sorry… non disclosure agreement ☹

Coverage Analysis


Code coverage of: 29.html
file:///Users/azaidman/IdeaProjects/C_TestCov
Google
Code coverage of: 29.html

C Test Coverage Calculator

Project name: ica
Module name: /ica/project/deelproject/TDFS/sources/tdfs_mut2.c
Date: Mon Aug 01 13:54:00 CEST 2005
Coverage: 95.23809523809523%
Approximated statement coverage: 89.94764397905759%

Up one level
Previous -- Next

main
CallControl
GetInfo
MakeMutlist
OpenMutlist
Write10Rec
Write20_40_80Rec
Write90Rec
NegativeCodedStrToInt
CreateDestin
CloseRemoveMutlist
OpenMut
CloseRemoveMut
ReadIndcijfers
ReadDemut1
GetDate
RemoveUndscore
RemoveNewline
CalcCC
System
bereken_modulus

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

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

	normal	with AOP
Build cycle	15 → 20 min	17h38m
Execute scenario	1h30m	+- 7h
Logfile size	90 GB (600 MB compressed)	
Code coverage	5h	
Frequency analysis	5h	
Webmining analysis	10h	

Submitted Papers

- *AOP for Legacy Environments, a Case Study, in European Interactive Workshop on Aspects in Software*, EIWAS '05 (Brussels).
- *Applying Dynamic Analysis in a Legacy Context: An Industrial Experience Report*, PCODA '05 (Pittsburgh, USA).
- *Aspect Orientation in the Procedural Context of C*, 6th FirW PhD Symposium, 2005 (Ghent).
- *Regaining Lost Knowledge through Dynamic Analysis and Aspect Orientation: An Industrial Experience Report*, CSMR '06 (Bari, It.).

Q&A