Secrets from the Monster:

Extracting Mozilla's Software Architecture

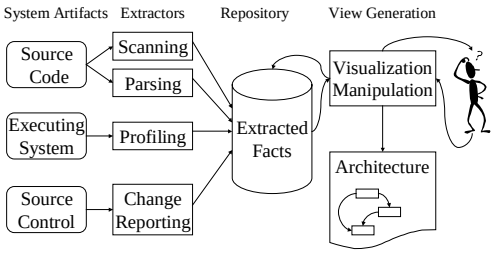
Michael W. Godfrey
Eric H. S. Lee

Software Architecture Group
Dept of Comp Sci, Univ of Waterloo

Background

- Reverse engineering tools can aid in recovering from "architectural drift"
  - RE tool: Fact extractor, manipulator, visualizer
  - Examples: PBS, Acacia, Rigi, TKSee, SHriMP, …
- Fact extractors vary in quality, detail, robustness, languages supported, …
  - Extractor interoperability has proven to be a huge headache
  - RE subtools often tightly coupled

Architectural Reconstruction



Motivation

1. Want to create architecture models of C++ systems, esp. Mozilla
   - Options: DIY … or … Gen++, Datrix, Acacia
2. Is there a "better" C extractor?
   - How do extractors compare quali/quantitatively?
3. Want to investigate data exchange between RE tools
   - WoSEF to be held tomorrow
4. (Later) want to build BEAGLE:
   - a tool for exploring program evolution

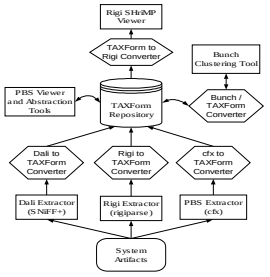
Extractor interoperability

- … just like western civilization
  - Researchers *want* this to work, tho
- Need to agree on
  - Syntax (TA, XML, SQL)
  - Semantic models (AST, CFG/DFG, SwArch)
- CoSET-99 paper:
  - TAXFORM suggested
  - Exploration of problems: unique naming, entity resolution, entity location (line numbers)
  - Preliminary case studies

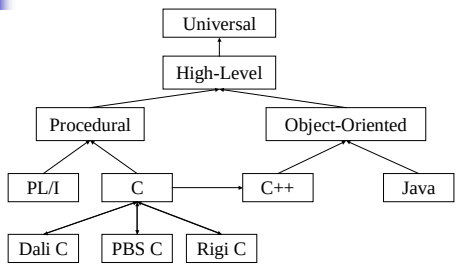
Exchange Format Reqs [CASCON '98]

- Support multiple source languages
- Scale to MLOC systems
- Provide mapping to source code
- Support static & dynamic dependencies
- Incremental approach
- Extensible, allowing new schemes to be defined as needed

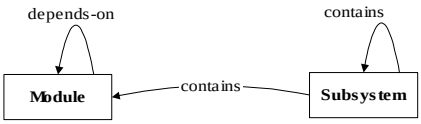
TAXForm Utopia



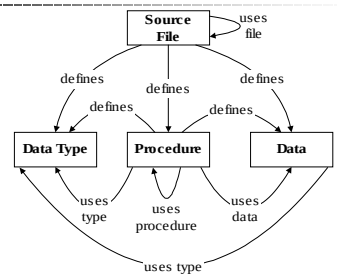
Transforming Between Schemas



TAXform — High level schema



TAXform — Procedural schema



"Facts": PBS vs. Acacia

- PBS produces output in TA
  - tuples describe attributes of program entities/relationships:

```
funcdcl read.h fileClose
funcdef read.c fileClose
linkcall fileClose getFileSize
```

- Acacia produces two ";" delimited plain-text DBs: `entity.db` and `relationship.db`
  - Use SQL-like queries to get raw text output:

```
cdef -u func - def=dec
cref -u - - m – file2='*.h'
```

Translation Nuts and Bolts

- Acacia C model close to PBS's
  - 1:1 relationship between most kinds of facts; translation via `awk` and `ksh` scripts
- ... but "`linkcall`" harder as
  - acacia already does resolution of "`f calls g`" to the function defs; `cfx` does resolution at a later stage
  - no transitive closure for "`includes`"

  Solution: simple `grok` program
- `Ccia` problems:
  - less robust on some C systems
  - generates multiple UIDs sometimes

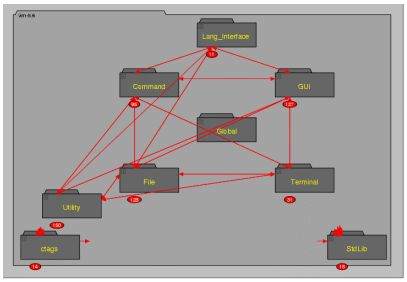
Guinea Pig #1: VIM text editor

- Examined VIM version 5.6:
  - 149 source files (.c, .h, .pro)
  - over 160 KLOC of K&R C
- Extraction results:

	Time (min:sec)	# facts
gcc compile	6:29	—
`cfx` extraction	4:27	43,000
`cia` extraction	1:52 + 3:20	51,000

  - Differences due to macro expansion, lib. var. refs, and missed fcn calls

Vim's architecture



Guinea Pig #2: Mozilla browser

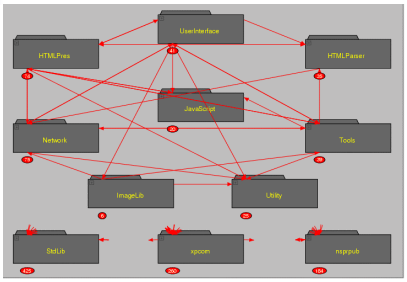
- "Open source" cousin of Netscape
- Examined Milestone 9 (M9):
  - Over 7400 files, 2 MLOC of C and C++
- Extraction results:

Full compile	0:35 hrs
Fact extraction (`Ccia`)	3:30 hrs
Fact manipulation (`grok`)	3:00 hrs
# of facts extracted	990,000

Mozilla extraction details

- Much extra work required:
  - Reconfigured PBS to understand OOPL schema
  - Complete rewrite of translation scripts (into `perl`) for efficiency
  - Some source code tweaking
  - More complex name mangling needed

Mozilla's architecture



Summary

- Created automated mechanisms for using the Acacia fact extractors within the PBS rev. eng. system
  - Tested on two large guinea pigs
- This work serves as an initial step towards data exchange between reverse engineering tools.
  - See proc. of WoSEF-00 for more discussion of this general topic.