ELEC 875
Design Recovery
and
Automated Evolution

Week 2 Class 1
Empirical Studies
Empirical Studies

• Studies Software Engineers
  ◊ what do they really do
  ◊ what do they really need
• Difficult
  ◊ multiple variables
  ◊ expensive
    - students/professional developers
    - real/artificial projects
  ◊ software engineers like other users are conditioned by their past
• Workshop on Empirical Studies of Software (WESS)
Lethbridge & Singer

• T.C. Lethbridge
  ◊ School of Information Technology and Engineering, University of Ottawa
• J. Singer
  ◊ National Research Council
• Study various companies in the Ottawa area
WESS ’97 Paper

• Understanding Software Maintenance Tools: Some Empirical Research
• Overview paper of Empirical Research
• What is a Tool
  ◊ Used by software engineer to perform a software engineering task
  ◊ hierarchical tools
• 5 Questions
  ◊ What tools and what tasks?
  ◊ What differences do tools make?
  ◊ Why use (or not use) a particular tool?
  ◊ What new tools or improvements to tools?
  ◊ How can tools be introduced to SEs?
**WESS ’97 Paper**

- Measures
  - What tools are used
  - Number of times each tool is used
  - Elapsed time spend using a tool
  - Goals and tasks for particular usage of a tool
  - List of positive attributes
  - List of negative attributes
  - Time to perform a given task
WESS ’97 Paper

• Data Collection
  ◇ Questionnaires (web based)
  ◇ Interviews
    - General structured interviews
      60-90 minutes, 10 page protocol (24 developers)
    - Regular debriefings (every few weeks)
      30-60 minutes
    - Tool reviews - specific tool and subtotals
      30-60 minutes
WESS ’97 Paper

- Data Collection
  ◊ Observation
    - real work (30 minute session)
    - use same tools and techniques?
    - artificial tasks

◊ Automated logging of tool use
WESS ’97 Paper

• Data Interpretation
  ◊ Small group of engineers
    - statistical sample?
    - defined process
    - domain specific (complex real-time software)
  ◊ larger and more diverse groups?
WESS '97 Paper

- Observations
  - Most used tools - editors
  - Second most used tools - searching tools
  - Explore software as much as edit software
  - Wish list - analysis tools
    - statistical sample?
WESS ’97 Paper

• Positive Features of Tools
  ◊ Ease of use
  ◊ Useful tools
  ◊ speed of tools

• Generic positive NF requirements
WESS ’97 Paper

- Negative Features of Tools
  - lack of integration
    - don’t want to manually transfer data between tools
  - wrong mix of features

- Difficulty introducing new tools
  - resistance to new tools
  - significant effort to learn new tool
    - will it be worth it?
  - train a single individual to act as consultant within user group
WESS ’97 Paper

• Future Studies
  ◊ collaboration between researchers (spread effort and cost)
  ◊ questionnaires and logging tools in more companies (contacts)
  ◊ observe different engineers in different environments using same tools
  ◊ interviews with different groups of SEs

• Issues
  ◊ Same questionnaires and interview protocols
  ◊ similar methodologies - training/experience/presentation
• Same Research, more Depth
  ◊ identifies some problems with traditional ESP approaches
  ◊ understanding how programmers solve problems does not necessarily lead to better tools

• Usability vs Useful
  ◊ Usability - clarity of interface
  ◊ done in an artificial environment
    - isolated from other factors
    - user forced to use tool
  ◊ does not guarantee that the software is useable
    - would he use the software
CASCON ’97 Paper

- Telecommunications company
  ◊ several million lines of code (16k funcs, 8k files)
  ◊ well defined process

- Survey
  ◊ Reading Documentation tops the list
  ◊ look at source
  ◊ design near bottom of list
  ◊ 57% of time fixing bugs, 35% of time making enhancements
    - differs from published norms, survey effect or difference in business?
  ◊ Validity of surveys?
• Individual Study
  ◊ new employee (experienced)
  ◊ weekly meetings at start
  ◊ 3 weeks apart later
  ◊ mental model of system
  ◊ tasks, “new” information
  ◊ shadow user, record activities
    - observer effect?
  ◊ search is most frequent activity
CASCON ’97 Paper

• Group Study
  ◊ mental model of system
  ◊ interviews
  ◊ shadow user, record activities
  ◊ looking at source, searching is most frequent activities
  ◊ reading docs low on list (although high on survey)
Company Study
◊ company uses custom tools
◊ tool group collects statistics on tool usage (tools log their usage)
◊ compiles - 41% most often
  - nightly builds
  - testing groups
  - excluded
◊ search most frequent activity
◊ editors low - why?
Results

◊ search seems to be where SEs spend most of their time
◊ improving search seems to present the greatest opportunity for support

Just In Time Comprehension

◊ system too large to comprehend
  - general understanding
  - task determines what is comprehended
  - ignore rest of problem
CASCON ’97 Paper

- Tool Functional Requirements
  - search for semantic entities in source code
  - display results of search and relationships
  - searches are repeated (history)
- Non-functional requirements
  - system size
  - performance
  - more than one language
  - interoperability
  - independent interfaces (research)
  - support JIC
CASCON ’97 Paper

- Problems with Existing Tools
  ◊ grep
    - no syntax or semantics
    - does not understand relationships
    - time
  ◊ editor searches
    - no semantics
  ◊ IDEs
    - more semantics, limited languages
    - eclipse?
  ◊ analysis tools
    - integration?
IDE

grep -i parse *.h *.cpp *.c

callback.h: * File: callback.h
callback.h:#ifndef CALLBACK_H
callback.h:#define CALLBACK_H
callback.h:struct callback {
callback.h:extern callback constraintcallback;
callback.h:#endif /* CALLBACK_H */
callback.cpp:#include "callback.h"
callback.cpp:callback constraintcallback = {

igmp.cpp:#include "callback.h"
igmp.cpp: //constraintcallback.QueryFn(tmp, header);

main.cpp:#include "callback.h"

rtps.cpp: // constraintcallback.RTPSMainModuleFn(tmp, header);

igmp.cpp:bool parseV2Report(PDU * thePDU, HeaderInfo * header, char *name) {
igmp.cpp: tmp->parseType = V2REPORT;

pmain.cpp: //parse the first (ethernet) header, grabbing the type field
pmain.cpp: //parse the IP header
pmain.cpp: parsedPDU = parseIGMPPacket(thePDU, header, argv[1]);

pmain.cpp: fprintf(stdout, "%Packets Parsed: %llu\nPackets Failed: %llu\nTotal Packets: %llu\nFailure rate: %0.2f%", count-failed, failed, count, ((float)failed/count) * 100);

ELEC 875 – Design Recovery and Automated Evolution
# IDE - CodeLite

```c
#include <stdio.h>

float AverageVal(int mst[], int n);
void calcDiff(int a1[], int a2[], int diff[], int n);

int main ()
{
    // declare and initialize the array
    int ice_mst[8] = {72, 78, 87, 94, 98, 101, 110}; // put values here
    int ice_mst2010[8] = {54, 62, 64, 69, 75, 80, 82, 84}; // put values here
    int diff[8], i;
    float avg_mst; //declare average variable here

    // call the function AverageVal to calculate and return
    // the average value
    avg_mst = AverageVal(ice_mst, 8);
    // print the value
    printf( "Avg mst 2005 : %0.2f\n", avg_mst);

    avg_mst = AverageVal(ice_mst2010, 8);
    // print the value
    printf( "Avg mst 2010 : %0.2f\n", avg_mst);

    calcDiff(ice_mst, ice_mst2010, diff, 8);
    printf("Differences\n");
    printf("Jan  Feb\n");
    for (i = 0; i < 4; i++)
        printf("%2d  %2d\n", diff[i], diff[i+4]);
}
```
IDE - Eclipse
CASCON ’97 Paper

• Problems with Existing Tools
  ◊ commercial browsing tools
    - sometimes no multiple languages (e.g. JNI)
    - some do support this
    - often limited integration
  ◊ academic
    - problems with integration, speed, automation