

**IEEE International Conference
on Software Maintenance**

Budapest, Sept 25-30, 2005



**Panel:
Identifications of Concepts,
Features and Concerns in
Source Code**

Andrian Marcus and Václav Rajlich

Wayne State University

Detroit, MI, USA

Notes 1

- Paolo – we need to be able to assess and compare methods before adding new ones to the table
- Vaclav: we need to combine techniques
- Harry: agree – must use various techniques – various objectives -> different approaches
- Rainer: we MUST! Dynamic analysis/feature location – fundamental misunderstanding: use `<>` execute

Notes 2

- Arie: Rainer is cynical “stairway to heaven”.
Involve the user from the beginning, the process should contain the user
- Vaclav: the user IS present already – narrow the search space for him/her. He still has to search in the end – we help the user
- Harry: the problem does not exist without the user. He is the start of the process.
- Rainer: do we need to know how the user work?
- Andi: yes we do

Notes 3

- Vaclav: study the reality vs. change the reality – we should be a reality based community
- Paolo: reaction to Arie – traceability links – how to address them during forward engineering – annotations are not effective – modularization is better, yet we do not have enough mechanisms
- Harry: the links have to be built into the development process – changing requirements should be traced into the source – traceability recovery too costly

Notes 4

- Arie: combine techniques with process
- Vaclav: “breaking” Arie – the index of a book is traceability – software is not a book, it is evolving- how do you keep links consistent?
- Arie: ...
- Rainer: I do not believe in fully automatic approaches.
- Arie seems to give up – agrees with Rainer 😊
- Vaclav: not optimist like Arie

Notes 5

- Paolo: cost effectiveness – big constraint.
Solution: modularization (e.g. aspects)
- Arie: problems with modularization – you can never do it completely
- Paolo: indeed modularization is not perfect
- Rainer: establishing link between feature and source include aspects (as they are represented in the source code)

Questions

- We need to modularize and re-modularize (reengineer)
- Vaclav: modularization = giving addresses to the source code parts; new structuring will not help evolution; we need to find things; how much modularization we need?
- Rainer: restructuring on demand – may or may not work, software has structure and it is not easy to change. Maybe the name software is a historical error!?!?

Questions

- Paolo: I do not see the limits of modularization, there is room for improvement – e.g. design patterns – success. However, marginal improvement, the limit is in sight
- Harry: modularity is not the answer. Granularity issue. Remember stepwise refinement? Shifted complexity at a lower granularity level (inside the module). Interactions between modules.
- Andrew (UBC): common sense should be used in choosing the best technique. Feature location tries to extract a new structure (modularization) ?!?! Is it right?

Questions

- Rainer: I don't see it. Feature location is only identifying a link from the concept to the source code, does not give a new structure.
- Q: Are we going to get to industry acceptance?
- Rainer: We should address that. My model is only from the point of view of research
- Orla: Used vs. executed. "I have a dream": IDE that allows not writing code when I have a feature and show how the code runs. Don't want to read through too much code. I need something useful and quick.

Questions

- Rainer: there are limitations. Technical details – see his paper. Need test cases – more than the usual ones. Goal is different here. We need to design these test cases for this goal.
- Vaclav: the techniques narrow down the search by 2 orders of magnitude – we need 3-4 orders of magnitude for large software (MLOC). Incremental improvement – combine tools and techniques
- Harry: the code should have links to the requirements, design, and test documents. This should be built into the IDE. Force the programmer to do it. The programmer is his own enemy – they do not want other to understand their code.

More questions

- Q. even if we have some documentation, we still can't find the relevant parts in the code – we need to analyze the language as well
- Q. ?
- Harry: It is not true now – adding annotations will not slow you down
- Arie: ?
- Vaclav: beware of programmers that write in “clever” way. Simple structure (understandable) = more maintainable.

Questions and comments

- Q. Do really the programmers have to protect themselves by not making the source understandable.
- Harry: IBM does it – they removed comments from source. Programmer are malicious. Programmer eats programmer.
- Q. most comments relate to the intention of the program. Open source is different – different goals – less malicious.
- Harry: OS community is the opposite
- Comment: perhaps the reaction to this “war”

Q&A

- Q. the bottom line in industry is not to produce code but to produce profit! Academia/research have to understand this
- Harry: motivation of programmers are driven by their environment
- Vaclav: we should have a paper on location concepts without comments (removed by the programmer)
- Harry: very difficult problem. Variable names help. But it can also hurt.

- Rainer: psychograms for programmers are different than for “normal” people. I do not think programmers are bad or malicious
- **www.cs.wayne.edu/~amarcus**