



Universidade
Federal da Bahia



Laboratório de
Engenharia de Software

On the Proactive and Interactive Visualization for Feature Evolution Comprehension: An Industrial Investigation

Renato Novais^{1,2}, Camila Nunes³, Caio Lima¹, Elder Cirilo³,
Francisco Dantas³, Alessandro Garcia³, Manoel Mendonça¹

¹Software Engineering Lab, Computer Science Department, Federal University of Bahia, Bahia, Brazil

²Information Technology Department, Federal Institute of Bahia, Campus Santo Amaro, Bahia, Brazil

³Opus Research Group, Software Engineering Lab, Informatics Department - PUC-Rio, Rio de Janeiro, Brazil
{renatoln, caiolima, mgmendonca}@dcc.ufba.br, {cnunes, ereioli, fneto, afgarcia}@inf.puc-rio.br

IM-UFBA, Av. Adhemar de Barros, s/n, Campus de Ondina, Salvador-Bahia-Brasil 40170-110
<http://les.dcc.ufba.br> (71) 3283-6343 les@dcc.ufba.br

Motivation



- It is widely recognized that program comprehension is a key aspect in software maintenance and evolution [1] [2].
- The comprehensibility of a program largely depends on the representation of its functionalities into logical units called *features*.



Motivation



- The maintenance and evolution of industrial software projects often requires the analysis of how one or more features evolved in the code history [1][2].
- A feature is a prominent or distinctive user-visible aspect, quality or characteristic of a software system [3].



Feature analysis



- Comprehension of features is primarily based on the identification of their implementation elements in the source code.
- As a result, tools have emerged to allow the mapping of the implementation elements realizing each feature. Some well-known examples of tools are:
 - FEAT [4] and ConcernMapper [5][6]: provide a lightweight visualization based on graph-based representation
 - SoQueT [7] that supports sort-specific queries
 - CIDE [8] that is based on an AST generic representation.
- **Unfortunately, they provide from none to very limited support for feature evolution analysis.**



Software evolution visualization



- There are also tools and approaches that aim to provide a graphical representation of the software evolution history,
 - Software visualization tools – such as Evolution Radar [9], Evolution Matrix [13], CodeCity [11], Moose [10], and CVSscan [12] – provide different views of the program modules' history.
 - diff tools [15][16][17] aim to detect the differences among the versions of an application in terms of added, modified and removed implementation elements.
- **Unfortunately, they are only able to capture and represent the evolution of the application modules (i.e. packages, classes, and methods).**



To tackle this problem...



- We present and assess a proactive and interactive visualization strategy to assist developers on feature evolution comprehension.
- In order to realize this strategy, we combined:
 - a set of heuristics [26][27] for proactively detecting feature evolution
 - extended our interactive differential visualization [18].
- We conducted a controlled experiment to evaluate the proposed visualization strategy.

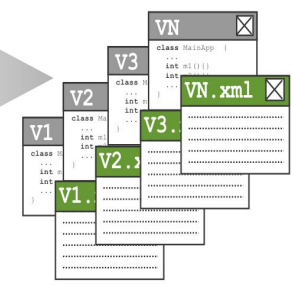
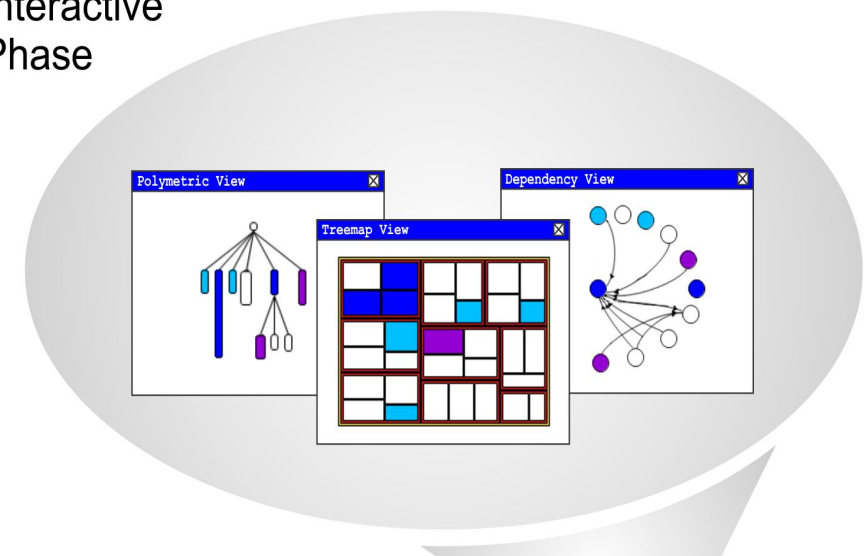
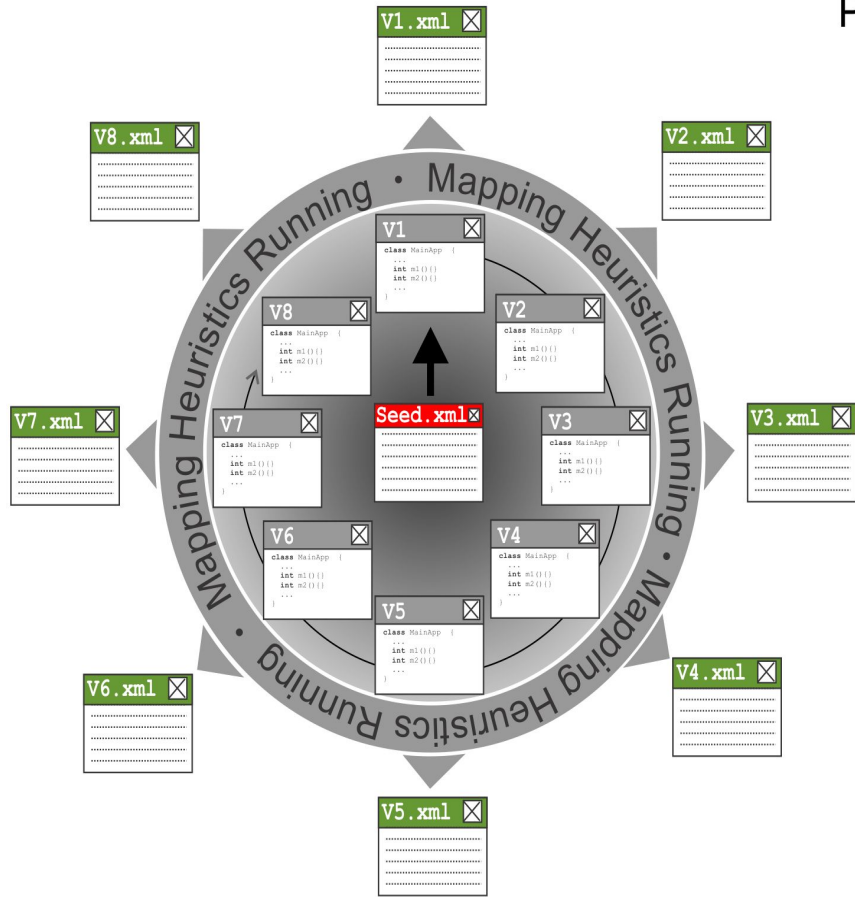





The Proactive and Interactive Visualization






Proactive Phase

Interactive Phase



-  Entire code of a Java application
-  Mapping Expansion
-  Seed

-  Added elements in feature code
-  Removed elements in feature code
-  Transferred elements from a feature code to another one

Proactive Phase



- The seed mapping contains an initial set of implementation elements realizing the features.
- There are a set of five mapping heuristics which are:
 - detecting omitted feature partitions,
 - detecting communicative features,
 - detecting code clone,
 - detecting interfaces and super-classes,
 - detecting omitted attributes.



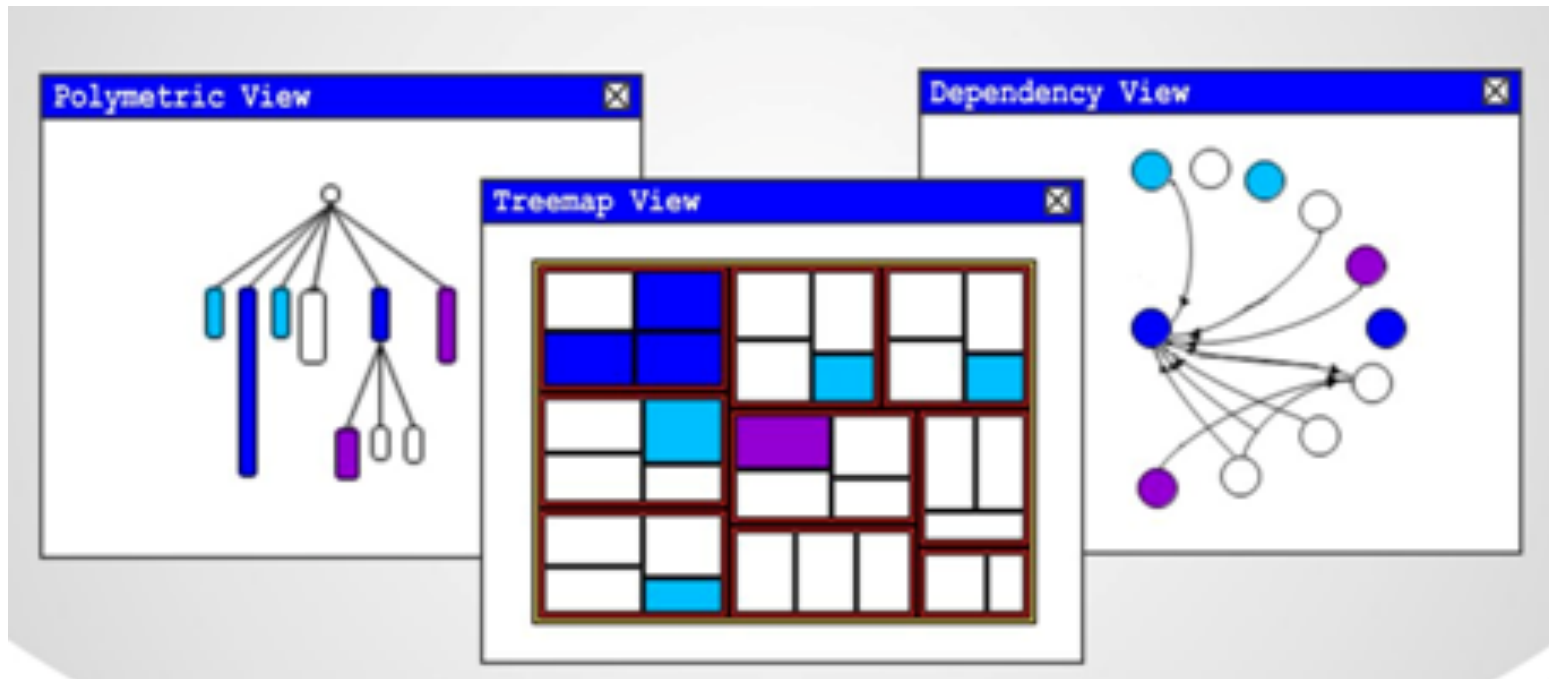
Interactive Phase



- SME is an integrated, interactive and coordinated multiple views environment.
- It uses three views to address the feature evolution comprehension.
- Each view provides means for analysing the feature evolution under different perspectives: structure, inheritance and dependency.



Views and colors used

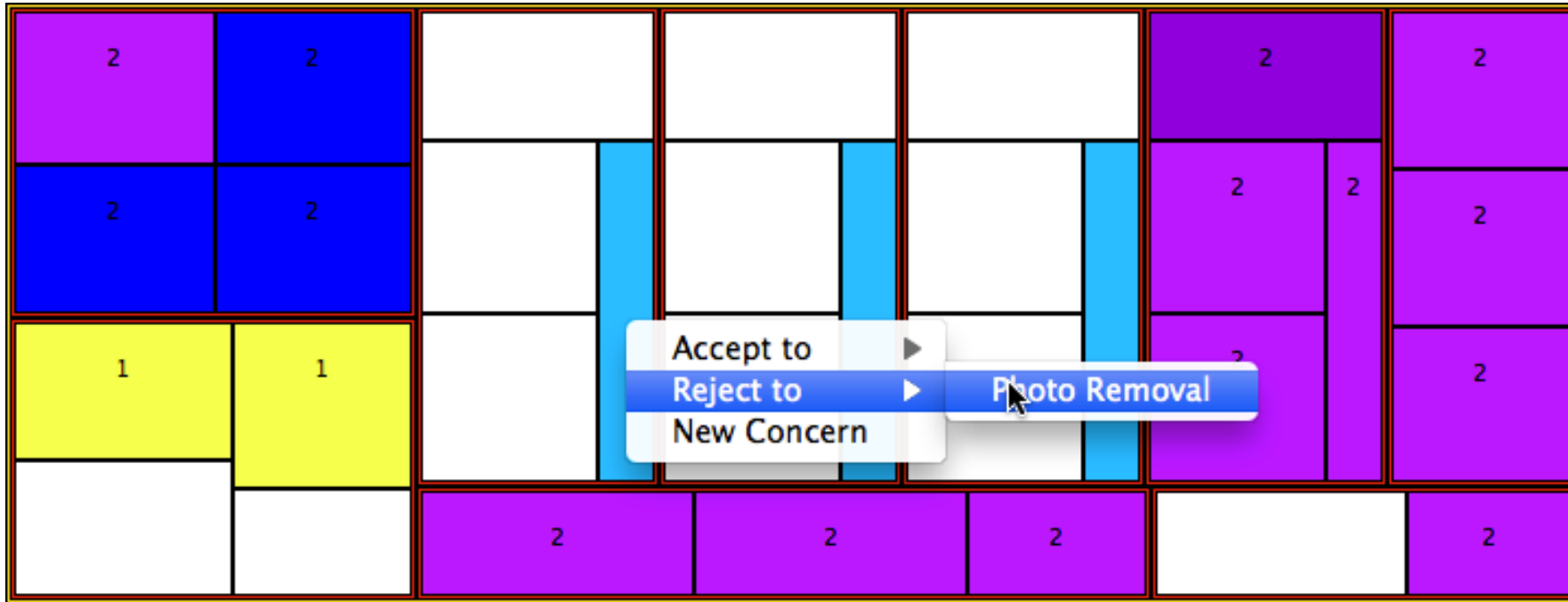


Interactions

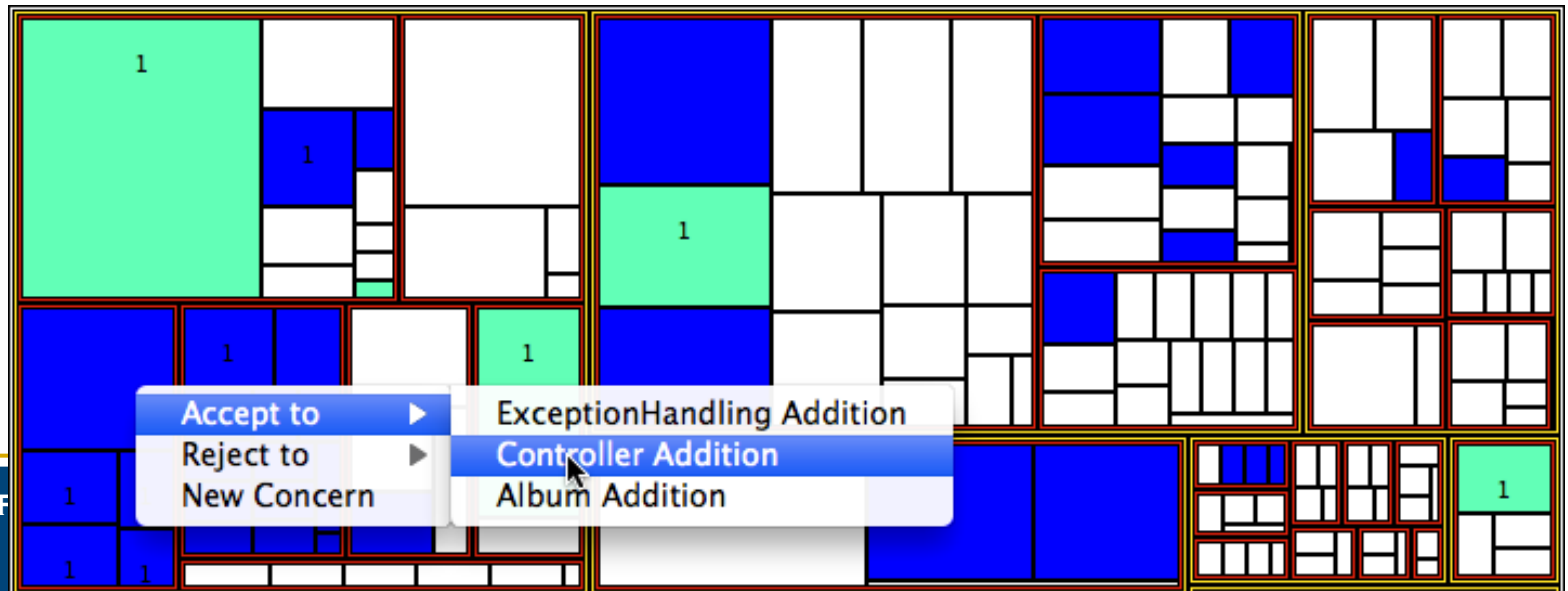


The screenshot displays three views of a software project named "MobileMedia04_OO" (P: 12 C: 31 M: 162):

- TreeMap:** A treemap visualization showing the distribution of code elements. The largest area is blue, with smaller yellow and white areas. Numbers (1 and 2) are placed within the rectangles to indicate counts or sizes.
- Polymetric:** A diagram showing vertical bars of varying heights and widths, representing different metrics or relationships between classes. Some bars are blue, some are yellow, and some are white.
- Dependency:** A graph showing class dependencies. Nodes are circles, and arrows indicate dependencies. A context menu is open over a node, listing actions: "Add to concern", "Remove from concern", "New Concern", "Open on TreeMap", "Labelling", "Sorting", "Controller", and "Album".



Accept or reject the heuristic suggestions





EXPERIMENTAL EVALUATION



SOFTWARE ENGINEERING LABORATORY – UFBA - <http://les.dcc.ufba.br>

RENATO NOVAIS - renatonovais@gmail.com

The experiment



- The goal of this experiment is to evaluate quantitatively the effectiveness of the visualization strategy when compared to a tree-structure strategy - ConcernMapper.
- Although diff tools [15][16][17] are more used than the CM tool in industry, we did choose the later (as baseline) for the following reasons:
 - diff tools do not provide explicit visualization support for features, the main concept addressed in this work
 - their purpose is completely different as they are intended to show the overall textual differences of the entire application.
- It is also important to remark that both the SME and CM tools are supported as plug-ins of the Eclipse IDE.



Study Hypothesis



- The effectiveness of the strategies is measured quantitatively in terms of time and correctness.

Hypotheses	Description
H1	The SME strategy decreases the time spent in feature evolution comprehension tasks.
H2	The SME strategy improves the correctness of feature evolution comprehension tasks.



Experimental Object



- This experimental object is a logistic software application for the oil industry.
- it is a real world and evolving application that has been developed since 2006;
- it has a significant size, on average 120 KLOC, and complex modules;
- it underwent various forms of changes during their evolution;
- it contains a significant set of functionalities with different degrees of granularity;
- its features have a large number of implementation elements realizing them.



Chosen Features



- We chose a few features from the application domain as well as a well-known crosscutting feature whose concept is widely known

Features	Descriptions
Notification	It defines a system notification to users (i.e., email).
Route	It represents a route of products between two points in the logistics context.
Report	It represents the report exhibition, exportation and printing.
Transaction	It is responsible for storing and recovering data from the database and ensuring ACID properties.



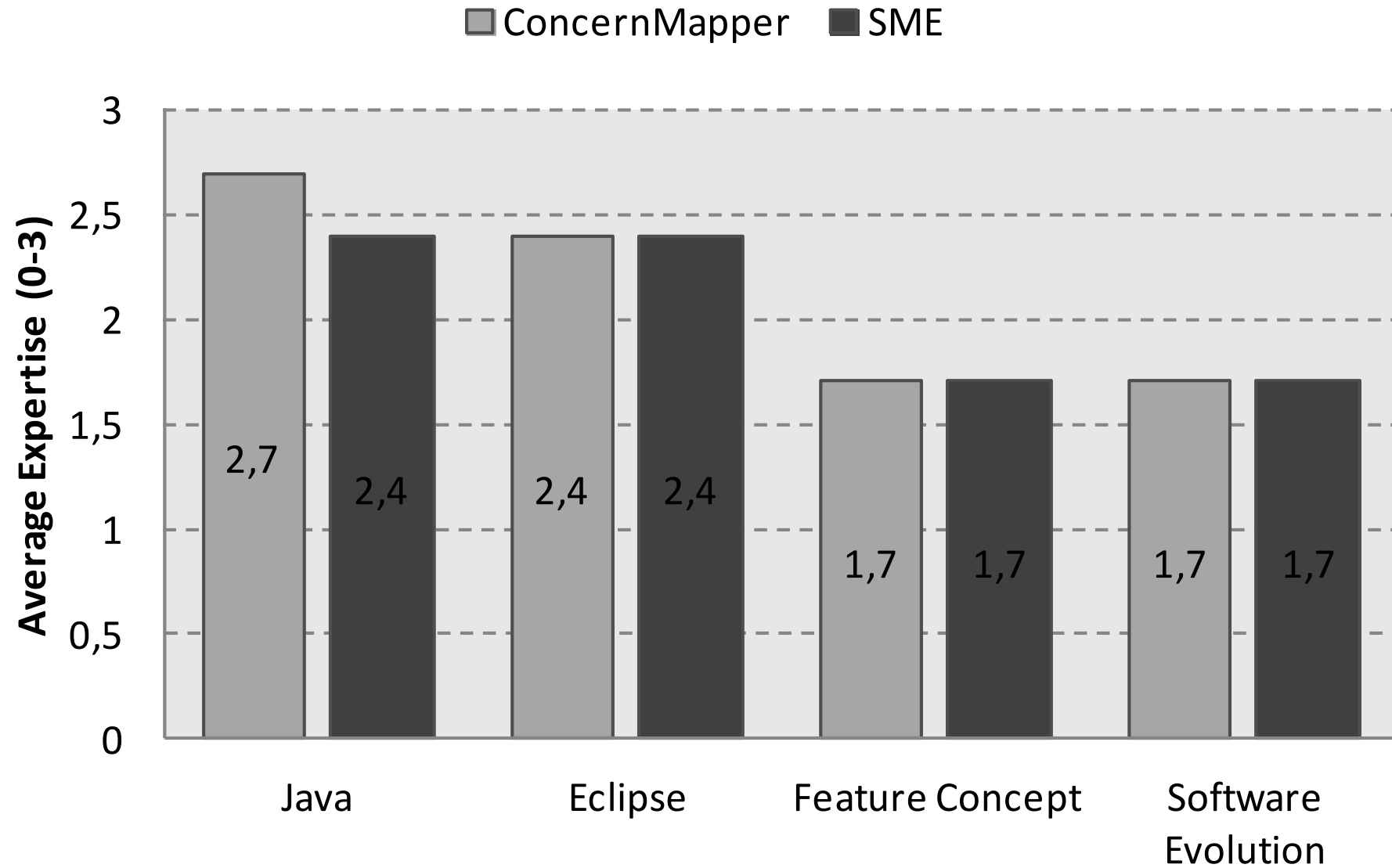
Participants



- We run the experiment with 20 participants who work in industry as software developers in two venues in Brazil (Salvador – BA and Rio de Janeiro – RJ).
- We selected experienced (more than five years) and novice developers (less than five years) to assure that the strategies can be beneficial for both cases.
- We used a characterization form available at [30]



Participants: CM x SME



Task Design



Tasks	Goals	Descriptions
T1-3	Feature Evolution Analysis	From version 1 to 2 what are the methods in the <code>ValidationFrame</code> class now realizing the feature <i>Notification</i> ?
		From version 2 to 3 what are the methods in the <code>Route</code> class not realizing the feature <i>Route</i> anymore?
		From version 4 to 5, what is the method that now realizing new features and not realizing old ones anymore? What are these features?
T4-5	Feature Tangling Analysis	What are all features realized by the <code>DBConnection</code> class in each version?
		Is the <code>ServerConfig</code> class realizing more than one feature through the application evolution? If yes, what are the features? For each feature, in each version, what are the methods realizing it?
T6	Feature Dependency Analysis	From version 1 to 2, what are the features of which the <code>RemoveFolderCall</code> class depends on? What are the features realized by the <code>RemoveFolderCall</code> class?



Experimental Procedures



- The CM and SME tools used the mappings generated by the heuristics
- The participants filled-in the characterization form (Section C) and were allocated to one treatment (strategy to be used during the comprehension tasks);
- A training session of 30 minutes was conducted in order to explain and show how both tools work.



Experimental Procedures



- A practice homework was sent to each participant to make them familiar with the tool of their treatment. We used the MobileMedia system [29] as experimental object.
- The experiment itself was run in university laboratories where the entire environment was set up.
- We analysed the results.
 - An entire task correctly answered was worth 1 point.
 - Wrong answers counted negatively.

Task	Worth
Correct	1
Partially correct	(0,1)
Incorrect	0



Experimental Procedures



- Finally, we applied a feedback questionnaire. This questionnaire was applied on-line in order to guarantee anonymity and avoid intimidating the participants.
 - They were asked about the experiment design, training, homework, execution, and tasks.



Pilot Study



- A pilot study was performed prior to the experiment with the intention of identifying certain problems in its procedures, or even in the tools, which are difficult to predict during its execution.
- Four participants were selected to perform the pilot. Two of them used SME and two of them used CM





RESULTS



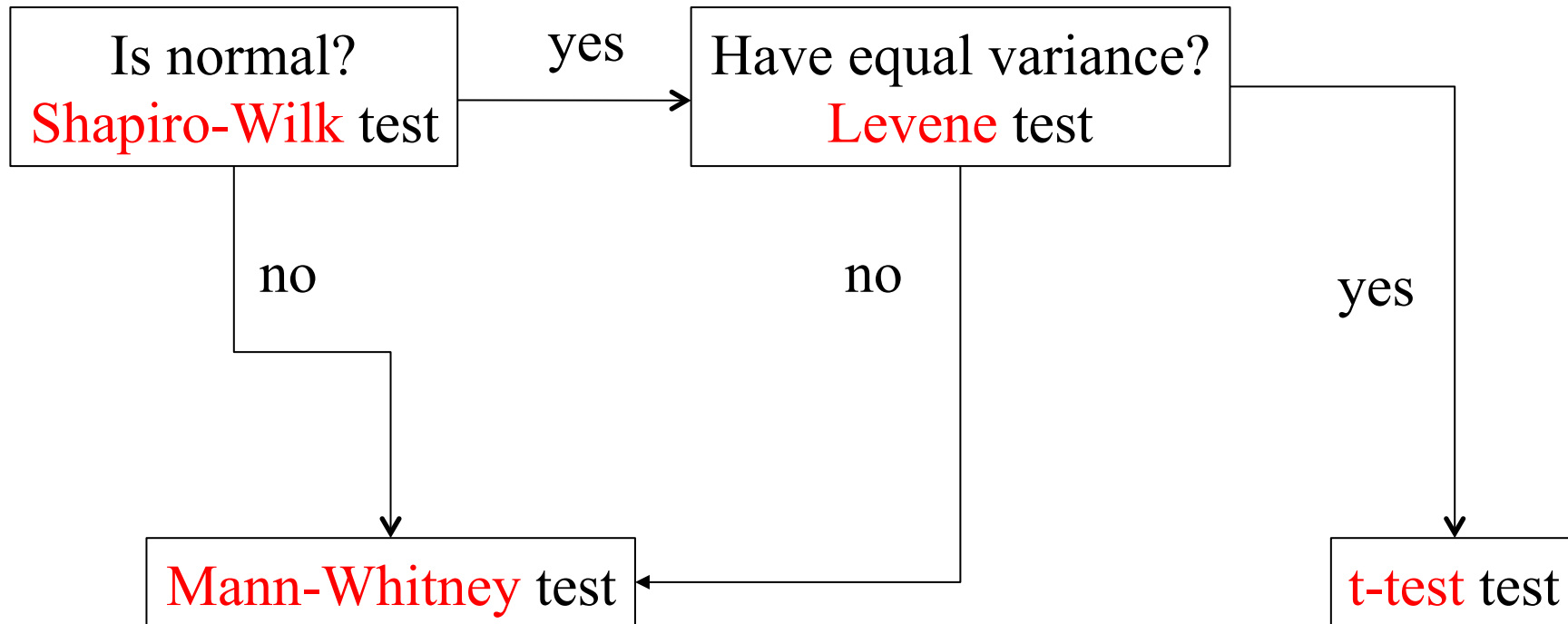
Statistical Test



- The statistical analysis was composed of
 - descriptive analysis was produced with the computation of measure's mean, minimum value, maximum value and standard deviation.
 - Verified if the data was normal and had equal variances
 - we selected two tests, t-test and Mann-Whitney, to be applied depending if the sample distributions were normal and homoscedastic, or not.
 - We run the tests
 - We used a confidence level of 95% ($\alpha=0.05$).



How to select the statistical test? (considering a simple design: one factor two treatments)



R-project



- Using R project (<http://www.r-project.org/>) to perform the tests

Correctness

SME

```
seva_acertos_to...
Arquivo  Editar  Formatar  Exibir
Ajuda
# SEVA all correctness
#
correct group_a
5.56 SEVA
6.00 SEVA
6.00 SEVA
5.22 SEVA
5.00 SEVA
5.11 SEVA
6.00 SEVA
4.00 SEVA
6.00 SEVA
6.00 SEVA
```

CM

```
seva_acertos_to...
Arquivo  Editar  Formatar  Exibir
Ajuda
# SEVA all correctness
#
correct group_a
4.89 TDA
2.83 TDA
4.80 TDA
4.80 TDA
3.89 TDA
4.00 TDA
3.54 TDA
4.83 TDA
4.66 TDA
5.00 TDA
```



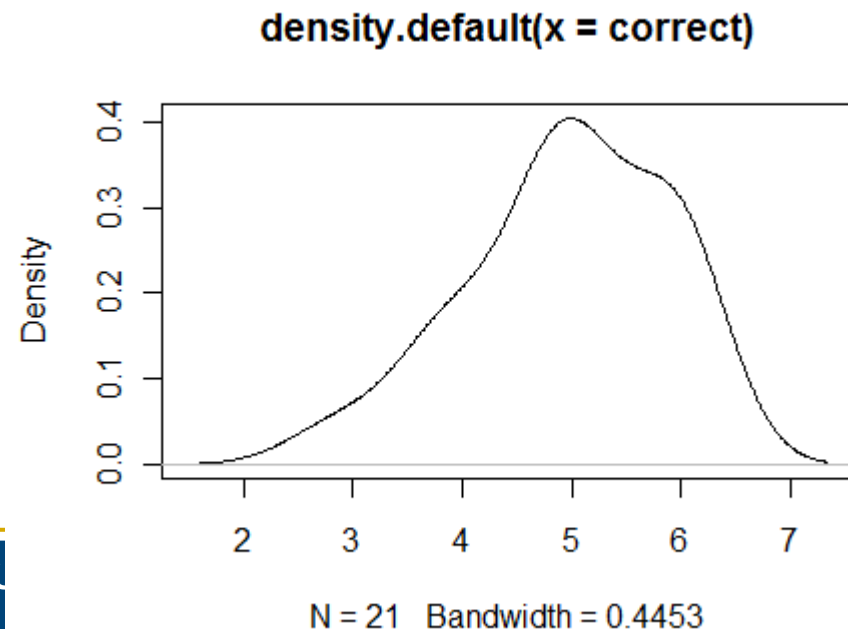
SOFTWARE

Normality in R-Project



- Draw the density graphic

- `acertos_total_ufba <- read.table("D:/RenatoNovais/dados/seva_acertos_total_ufba.txt", header=TRUE)`
- `correct = acertos_total_ufba$correct`
- `plot(density(correct))`



Normality in R-Project



- Using Shapiro-Wilk test
 - `tda_c <- c(4.89, 2.83, 4.80, 4.80, 3.89, 4, 3.54, 4.83, 4.66, 5)`
 - `shapiro.test(tda_c)`
 - `sme_c <- c(5.56, 6, 6, 5.22, 5, 5.11, 6, 4, 6, 6)`
 - `shapiro.test(sme_c)`



Variance in R-Project



- *Defining Levene*

```
> levene.test = function(data1, data2) {  
+   levene.trans = function(data) {  
+     ## find median for group of data  
+     ## subtract median; take absolute value  
+     a = sort(abs(data - median(data)))  
+     ## if odd sample size, remove exactly one zero  
+     if (length(a)%2)  
+       a[a != 0 | duplicated(a)]  
+     else a  
+   }  
+   ## perform two-independent sample T-test on transformed data  
+   t.test(levene.trans(data1), levene.trans(data2), var.equal = TRUE)  
+ }
```

- Using Levene test: *levene.test(tda_c, sme_c)*



Hypotheses' tests



- Mann-Whitney test

- `task_acertos_total_ufba_puc <- read.table("D:/RenatoNovais/dados/seva_acertos_total_ufba_puc.txt", header=TRUE)`
- `attach(task_acertos_total_ufba_puc)`
- `wilcox.test(correct ~ group_a, data=task_acertos_total_ufba_puc)`

- t-test

- `t.test(correct~group_a, alternative="two.sided", var.equal=TRUE, conf.level=.95)`



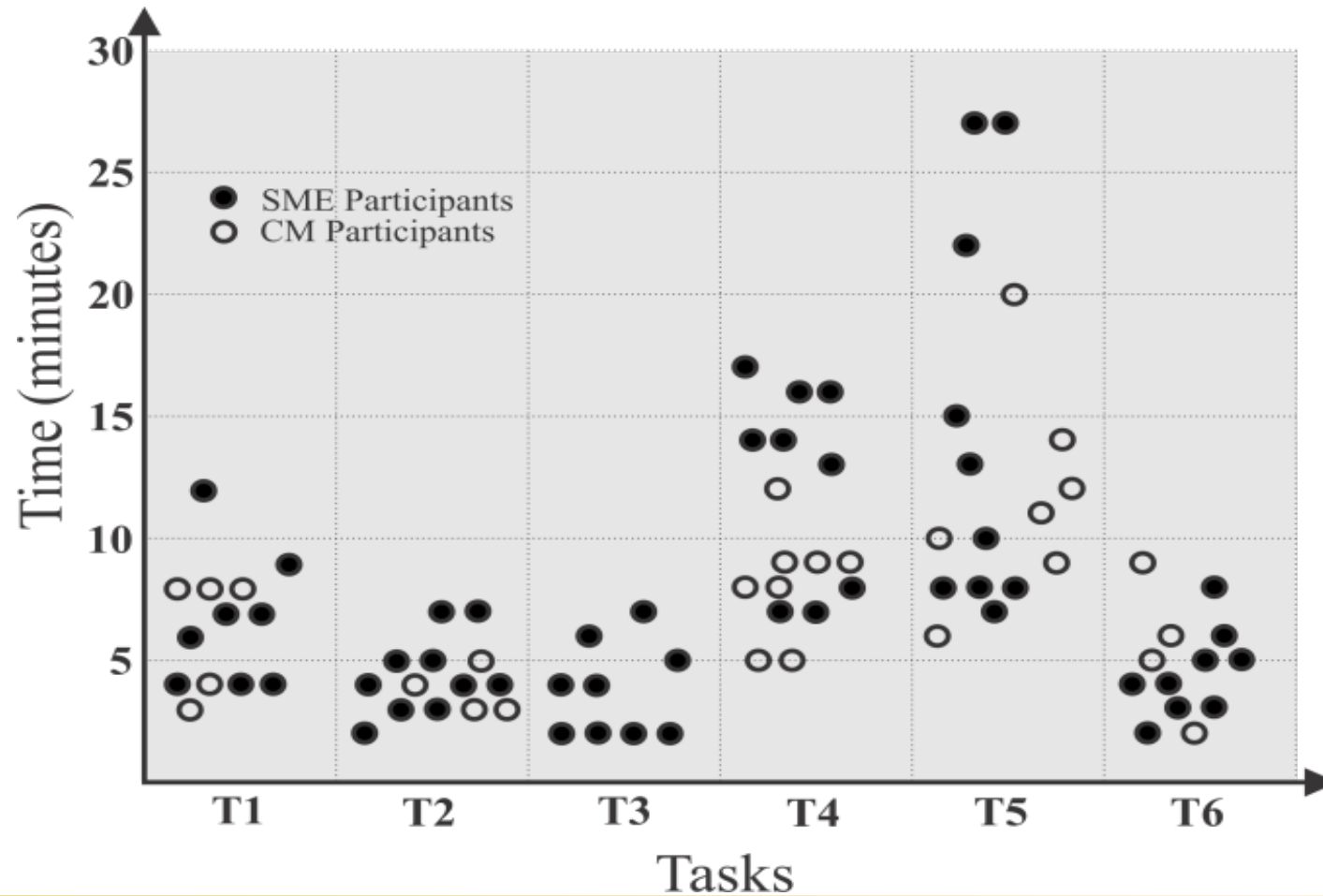
DESCRIPTIVE STATISTIC OF THE EXPERIMENT RESULTS



	Group	Mean	Diff	Min	Max	StdDev	S-W p-value	Levene	t-test p-value	MW p-value
Time (minutes)	CM	52.30		28.00	79	13.38	0.83340			
	SME	47.45	- 8.95%	29.00	62	9.02	0.81460	0.3963	0.3754	
Correctness (points)	CM	4.32		2.83	5	0.37	0.04324			
	SME	5.49	+ 26.94%	4.00	6	0.67	0.01297			0.001192



H1: Time Spent



Tool Configuration



- Using CM, the participants only need to open the mapping files of each version in its view and observe the differences.
- Using SME, the user needs to specify the versions to be compared, select colors for the analyzed features, and open a view to analyze them.
- **We observed that when the tasks do not demand much effort, the configuration time significantly influences the final results (Task T1).**



Feature Analysis Issues



- This factor is related to the time used for observing the feature differences (removed, transferred and added) across the versions.
- CM participants use of set theory for identifying, for instance, which elements were removed.
- SME participants use colors to identify the feature's implementation elements.
- **the use of SME yielded faster performance of the participants while successfully accomplishing their tasks (tasks T2, T3, T6)**



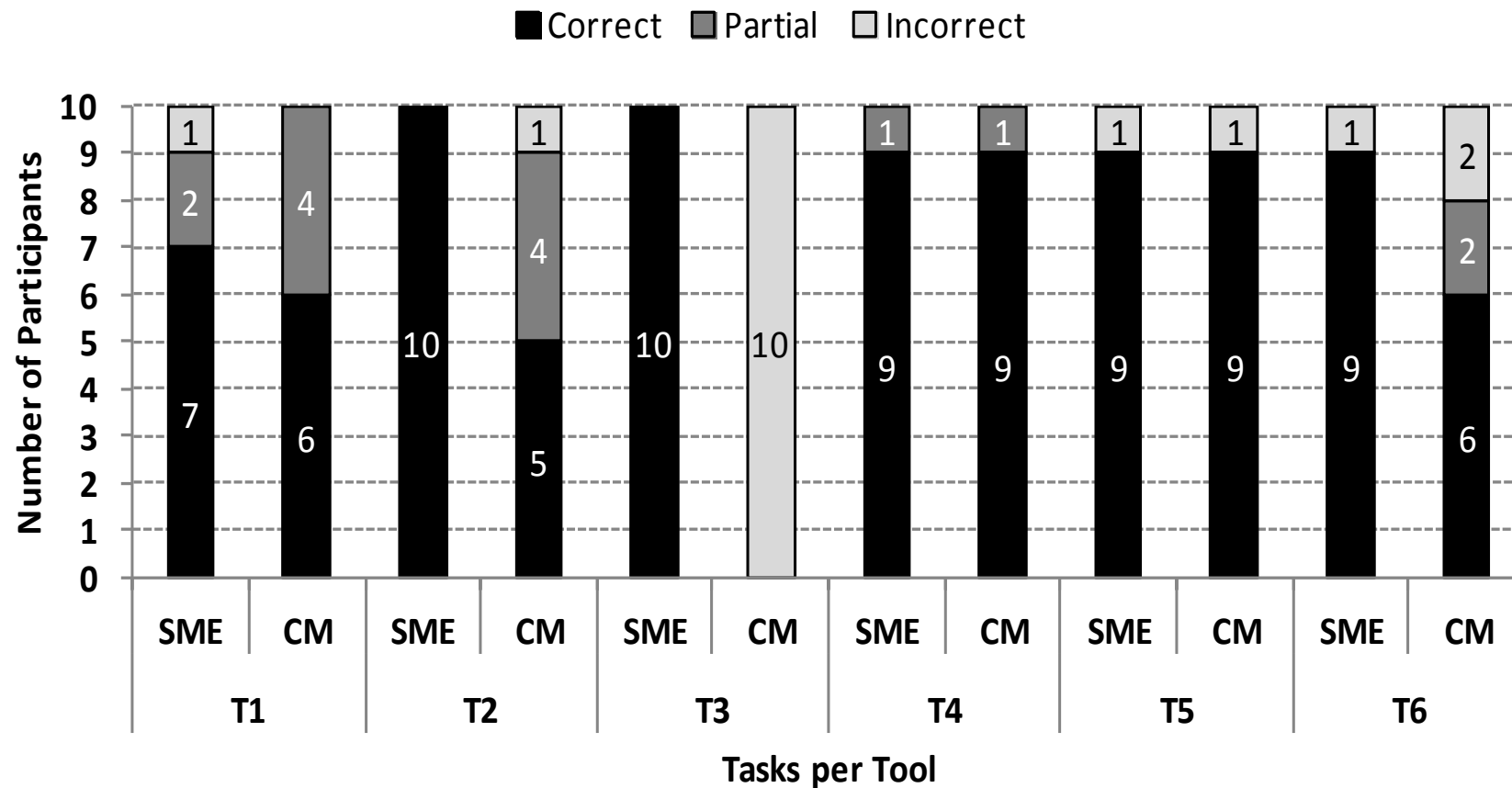
Tool Processing cost



- The processing cost refers to the time that a tool takes for executing a participant's action
- for processing T4 and T5, SME demanded more time due to the graphical representation of the large data.



H2: Correctness



Feature Visualization



- SME provided solutions with higher degree of correctness when compared to CM. (T1, T2, T3 and T6).
- The SME superiority mainly occurred due to the availability of several views that helped the participants to adjust or confirm their answers from different perspectives
- the use of colors also helped to clearly distinguish the removed, transferred and added actions performed on features' elements



Evolution Analysis



- The number of incorrect answers increases when the feature evolution analysis becomes more complex.
- In task T3, all participants who used SME provided correct answers while those who used CM completely failed in providing correct answers.



Proactive Mappings



- It is realistic to expect that such mappings may contain a few imperfections or mistakes as those ones performed by hand
- For the minor mapping mistakes, we observed that the use of visualization can help to circumvent them.
- Reject or accept the heuristics





LESSONS LEARNED



Time and Correctness can go hand-in-hand



- it is possible to reach a higher degree of correctness when there are distinct feature representations.
- Even considering that some participants took more time to provide answers, 90% of them successfully accomplished the tasks using SME
- The SME tool could be satisfactorily used by experienced and novice participants [30].



User's Point of View



- the tools should provide visual resources in order to facilitate smooth analyses of source code evolution
- Double check their answers
 - Using colors
 - Menu pop-up





THREATS TO VALIDITY



Conclusion Validity



- ***design of the tasks***: the tasks of feature evolution comprehension could have been too difficult or biased to a specific tool
 - Pilot study
- ***time restrictions***: the time allocated to the experiment could have influenced the participants' answer
 - *Pilot study*
- ***heterogeneity of the participants***: it refers to the selection of the participants involved in the experiment
 - it contributes to minimize the external threats of the study



Construct Validity



- *operational procedures of the experiment*: the participants may have not properly understood the experiment guidelines.
 - Training and homework
- *confounding constructs and levels of constructs*: it refers mostly to the expertise level of the participants.
 - Participants with different levels of experience



Internal Validity



- *allocation of participants to the treatments*: it is related to the partition of the participants in the groups (CM and SME).
 - Used Characterization form answers in order to make the partition as fair as possible



External Validity



- *representativeness of the artefact used*
 - we selected five versions of an evolving industrial application, which is quite representative in terms of changes and size.





CONCLUSIONS



Conclusion



- We presented a proactive and interactive visualization strategy for assisting developers in feature evolution comprehension
- We performed an experiment to evaluate the effectiveness of SME
- These initial results demonstrated the benefits of SME and its usefulness to help developers in feature evolution comprehension mainly regarding correctness.





Universidade
Federal da Bahia



Laboratório de
Engenharia de Software

On the Proactive and Interactive Visualization for Feature Evolution Comprehension: An Industrial Investigation

Renato Novais^{1,2}, Camila Nunes³, Caio Lima¹, Elder Cirilo³,
Francisco Dantas³, Alessandro Garcia³, Manoel Mendonça¹

¹Software Engineering Lab, Computer Science Department, Federal University of Bahia, Bahia, Brazil

²Information Technology Department, Federal Institute of Bahia, Campus Santo Amaro, Bahia, Brazil

³Opus Research Group, Software Engineering Lab, Informatics Department - PUC-Rio, Rio de Janeiro, Brazil
{renatoln, caiolima, mgmendonca}@dcc.ufba.br, {cnunes, ereioli, fneto, afgarcia}@inf.puc-rio.br

IM-UFBA, Av. Adhemar de Barros, s/n, Campus de Ondina, Salvador-Bahia-Brasil 40170-110
<http://les.dcc.ufba.br> (71) 3283-6343 les@dcc.ufba.br

References



- [1] T. A. Corbi, “Program understanding: Challenge for the 1990s,” IBM Systems Journal, vol. 28, no. 2, pp. 294–306, 1989.
- [2] H. Bennett and V. Rajlich, “Software maintenance and evolution: a roadmap,” Proc. of the ICSE, ACM, NY, pp. 73-87, June 2000.
- [3] K. Kang, et al., “Feature-oriented domain analysis (foda) feasibility study,” Tech. Report, Carnegie-Mellon University Software Engineering Institute, November 1990.
- [4] M. Robillard and G. Murphy, “Representing concerns in source code,” ACM Trans. Softw. Eng. Meth. 16, 1, February 2007.
- [5] M. Robillard and G. Murphy, “Concern graphs: Finding and describing concerns using structural program dependencies,” Proc. of the ICSE, ACM: NY, USA, 406–416, 2002.
- [6] M. Robillard and F. Weigand-Warr, “Concernmapper: simple view-based separation of scattered concerns,” Proc. of the OOPSLA workshop on Eclipse Technology, ACM, NY, USA, pp. 65–69, 2005.



References



- [7] M. Marin, et al., “SoQueT: query-based documentation of crosscutting concerns,” *Proc. of the ICSE, USA*, pp. 758-761, 2007.
- [8] J. Feigenspan, et al., “Using background colors to support program comprehension in software product lines,” *Proc. of the EASE, Durham, UK*, pp. 66-75, April 2011.
- [9] M. D'Ambros, M. Lanza and M. Lungu, “Visualizing co-change information with the evolution radar,” *IEEE Trans. Softw. Eng.* vol. 35, no 5, pp. 720-735, September 2009.
- [10] S. Ducasse, et al., “Moose: a collaborative reengineering environment,” *Proc. of the Tools for Software Maintenance and Reengineering, Liguori, Napoli, Italy*, pp. 55-71, 2005.
- [11] R. Wettel, et al., “Software systems as cities: a controlled experiment,” *Proc. of the ICSE, ACM, NY, USA*, pp. 551-560, 2011.
- [12] L. Voinea, et al., “CVSscan: visualization of code evolution,” *Proc. of the SoftVis, ACM, NY, USA*, pp. 47-56, 2005.



References



- [13] M. Lanza, “The evolution matrix: recovering software evolution using software visualization techniques,” Proc. of the IWPSE, ACM, NY, pp. 37-42, 2001.
- [14] M. Lanza and S. Ducasse. “Polymetric views - a lightweight visual approach to reverse engineering,” *IEEE Trans. Softw. Eng.* 29, 9, pp. 782-795, September 2003.
- [15] M. Pilato, Version Control with Subversion. Sebastopol, CA: O'Reilly& Associates, Inc., 2004.
- [16] P. Cederqvist, Version management with CVS. Signum Support AB, 1993.
- [17] IBM. Rational Team Concert, “<https://jazz.net/projects/rational-team-concert/>,” Accessed in October, 2011.
- [18] R. Novais et al., “On the use of software visualization to analyze software evolution – an interactive differential approach,” Proc. of the ICEIS, pp. 15-24, 2011.



References



- [19] L. Hattori, et al. “Software Evolution Comprehension: Replay to the Rescue,” Proc. of the ICPC, pp. 161–170, June 2011.
- [20] T. Biggerstaff, B. Mitbender and D. Webster, “Program understanding and the concept assignment problem,” Communications of the ACM, vol. 37, pp. 72–82, May 1994.
- [21] T. Eisenbarth et al., “Locating features in source code,” IEEE Trans. Softw. Eng. vol. 29, pp. 210–224, March 2003.
- [22] D. Poshyvanyk, et al., “Feature location using probabilistic ranking of methods based on execution scenarios and information retrieval,” IEEE Trans. Softw. Eng, vol. 33, pp. 420–432, June 2007.
- [23] J. Buckner, et al., “Jripples: A tool for program comprehension during incremental change,” Proc. of the IWPC, IEEE Computer Society: Washington, USA, pp.149–152, 2005.
- [24] B. *Cornelissen, et al.*, “A controlled experiment for program comprehension through trace visualization,” *IEEE Trans. Softw. Eng, vol. 37, no. 3, pp. 341-355, June, 2011.*



References



- [25] T. Savage, et al., “Flat3: feature location and textual tracing tool,” Proc. of the ICSE, ACM, NY, USA, 2010, pp. 255–258, 2010.
- [26] C. Nunes, “On the proactive identification of mistakes on concern mapping tasks,” *Proc. AOSD*. ACM, NY, USA, pp. 85-86, 2011.
- [27] C. Nunes, et al., “Expansion of feature mappings in evolving program families: heuristic-based assessment,” Submitted to JSME, 2011.
- [28] B. Johnson and B. Shneiderman, “Tree-Maps: a space-filling approach to the visualization of hierarchical information structures,” Proc. of the VIS, IEEE Computer Society Press, pp. 284-291, 1991.
- [29] E. Figueiredo, et al., “Evolving software product lines with aspects: an empirical study on design stability,” Proc. of the ICSE, ACM, NY, USA, pp. 261–270, 2008.
- [30] R. Novais, et al., “On the Proactive and Interactive Visualization for Feature Evolution Comprehension: An Industrial Investigation,” <http://wiki.dcc.ufba.br/SoftVis/SMEFeature>, Accessed in October, 2011.



References



- [31] C. Wohlin, et al., Experimentation in software engineering: an introduction. Kluwer Academic Publishers: Norwell, USA, 2000.
- [32] C. Nunes, et al., “Revealing mistakes in concern mapping tasks: an experimental evaluation,” Proc. of the CSMR, IEEE Computer Society, Washington, DC, USA, pp. 101-110, 2011.
- [33] T. Nguyen, et al., “Aspect recommendation for evolving software,”. Proc. of the ICSE, ACM, NY, USA, pp. 361–370, 2011.

