Software Reverse Engineering Tool for Object Oriented Programming

D.M.Thakore
Department of Computer Engineering
Bharati vidyapeeth Deemed University
College of Engineering, Pune-43,
Maharashtra, India

S.J.Sarde (M.Tech Student)
Department of Computer Engineering
Bharati vidyapeeth Deemed University
College of Engineering, Pune-43,
Maharashtra, India

ABSTRACT

Software quality metrics plays the vital role in the process of assessing software as in qualitative and quantitative terms. Basically, software quality metrics determines specific but some important properties, attributes or characteristics of software in terms of numbers, values or some symbols. This type of assessment should be occurred according the some well-defined measurement rules. Software quality metrics are not only the static measurement state of project but also it will help in assessing the behavior, size, quality and complexity etc, of software. Evaluation of software quality metrics is used to predict the fault-prone area and components of software in early stage of reengineering process of existing software as quality indicators. These software quality metrics helps to identify the problem from software in early stages of the reengineering of existing software. Therefore, this paper deals with automate software quality metrics tool namely Software Reverse Engineering Tool (SRET) which should be developed to determine the different software quality metrics and attributes of object oriented programming. Hence these metrics measures the complexity, effectiveness, efficiency of software and these metrics should be save time and cost of software analyzer, developer, tester etc., for reengineering the existing software with less effort.

Keywords

Cohesion, complexity, object oriented design metrics, object oriented language, software engineering, software quality, software reengineering etc.

1. INTRODUCTION

The main objective of Reverse Engineering is to increase contact, awareness and understanding of existing software. It should be the first step in transferring effective software reengineering processes, methods, and tools into practical use. It should provide complete picture of the existing system and what system must do to satisfy the developer requirement and needs for software improvement, this is accomplished by constructing several models.

But the quality of software systems heavily depends on their structure, which affects maintainability and readability. However, the ability of humans to deal with the complexity of large software systems is limited.

Software metrics are used to predict critical information about software systems. Object oriented software development requires a different approach to software metrics. Software engineers generally use indirect measures that lead to metrics which provide a quantitative basis for understanding the underlying information in software development processes. Software metrics have always been important for software developers to assure the quality of some representation of software and organizations are achieving promising results through their use. The quality of software must be defined in terms that are meaningful to the users. Generally, quality objectives may be listed as performance, reliability, availability and maintainability, which are closely related to software complexity. Recovering this type of metrics is the first step towards reengineering a software system.

Therefore, Quality metrics are most important factors in software development. Quality based software improvement is essential for the software process improvement. Quality recognition mechanism and individual Quality metrics are suggested to avoid the large effort in typical measurement of quality in project.

Thus for above case there is no such automate tool or techniques which will find out model for the system which determine various software quality metrics specification.

Proposed system methodology approach should to measuring coupling, cohesion (Complexity), Data Access and Operation Access Metric (Security) relies on the assumption that the attributes, methods, relationship and classes of Object-Oriented systems are connected in more than one way.

2. LITERATURE SURVEY

Quality of software is very important into the computer science because quality is nothing but future requirements characteristics of end user or customer that is in measurable form. Basically, quality can be those product characteristics which meet end users requirements and thereby, provide product satisfaction. Hence, it is concluded that quality should be contain all properties (or characteristics or features) and vital attributes of a product which should satisfy the given requirements [1]. There are various standard definition of quality-1) quality contains all characteristics and significant features of a product or an activity which related to the satisfying of given requirements, 2) quality is nothing but totality of features and characteristics of product or a service that bears on its ability in satisfy the given need, 3) conformation to requirements, 4)meeting user requirements etc.[3] The important role of software process improvement is evaluating the current status of the software and decides the improvement proprieties. Therefore, it should the focus on the software process improvement that requires the need of software measure i.e. measurement of software quality metrics [3]. A software quality metric is a measurement of characteristics (or properties) of a software or its specification. So, quantitative measurements are essential in reengineering of

existing software. The purpose is gaining objective, goals, reproducible and quantible measurement which should be in numerical form, valuable in planning, schedule, debugging, software performance optimization etc. This is reason for the need of evaluating of software quality metrics of existing software system (as part of reengineering process) should be required when organization (or company) is decided to make changes into existing software system. The develop system which should automate and comprehensive tool which evaluate the software wide metrics, module wide metrics, attributes and hierarchical metrics for object oriented program by analyzing the source code of software. Software quality metrics can be classified into three categories-1) product, 2) process, and 3) project metrics out of these classification let focus on the product metrics which describes the characteristics of product such as size, complexity, design features, performance, security etc. Hence, develop a new automate software reverse engineering tool (SRET) which should analyze and shows as a new technique of getting software metrics from source code of software system i.e. deals with extracting software metrics from .class file of java programming.

3. OUTLINE

This paper is organized as follows. Section IV presents the related work. Section V represents various metrics and results. Section VI and VII presents concludes the paper by presenting results, conclusion and future work.

4. RELATED WORK

Proposed approach should evaluate software quality metrics for object oriented program like Java, software quality metrics are coupling, cohesion (Complexity), Data Access and Operation Access Metric (Security) relies on the assumption that the attributes, methods, relationship and classes of Object-Oriented systems are connected in more than one way. Complexity: Coupling: Low value->Good and High->Bad and cohesion is vice versa of coupling. Security: SDAM and OPM: Low value ->Bad and High value->Good

So, proposed approach should be minimize coupling and maximize cohesion for reduction in complexity of object oriented programming language , DAM and OPM metrics should much less as possible for greater security.

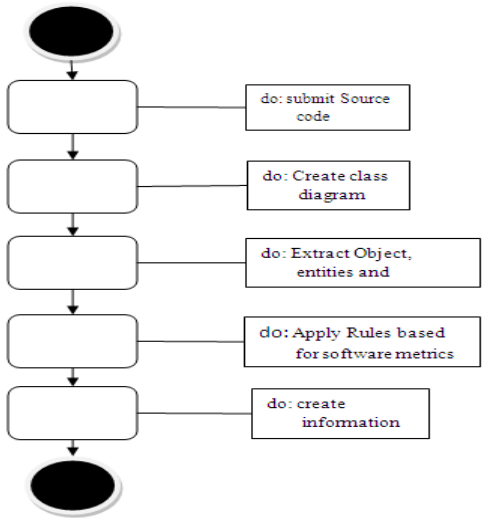

Fig 1: Activity Diagram for Software Reverse Engineering Tool

Proposed approach should use some better packages which help into the assessment of software quality metrics. These packages are parse the .class file of Java program and easily find out metric value.

5. METRICS DESCRIPTION AND RESULTS

Consider the example of Shopping Cart for understanding the assessment of software quality metrics from source code of .class file which represents the class relationship as shown below in figure.

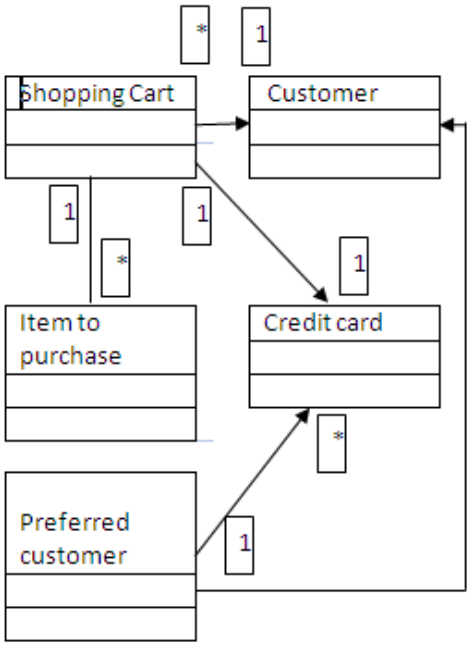

Fig 2: Example of Shopping Cart

1. Weighted Methods per class(WMC)

WMC is a software quality metric which assess the size and complexity. It is defined and represented as WMC = Σc_i i.e summation of no. methods of c_i where c_i is the complexity of all various methods numbered as c_1, c_2, … c_n in class C. If all methods in the class were assigned a complexity equal to 1 then the WMC for class C would corresponding the number of methods in the class. The no. of methods and the complexity of the methods concerned is a forecaster of how much time and effort is required to construct up and maintain the class. The greater the no. of methods in a class, the greater the potential impact on children; children inherit all of the methods defined in the parent class. Classes with large numbers of methods are likely to be more application specific, limiting the possibility of reuse.

Example: WMC is calculated by counting the number of methods in each class, therefore:

WMC for Shopping Cart = 2

WMC for Credit Card = 1

Table 1 . Summary Statistics of WMC

Metric	Max. Value	Min. Value	Avg. Value
WMC	27	2	14.5

2. Depth of Inheritance Tree (DIT)

Depth of Inheritance Tree (DIT) can be defined as depth of a class within the inheritance hierarchy or just like tree representation is the maximum number of steps from the class node to the root of the tree or measure no. of nodes from leaf node to root node of tree and is calculated by the number of ancestor classes. The leaf a class within the hierarchy has maximum no. of ancestor classes then the maximum the number of methods it is likely to inherit making it more complex to forecast its behavior. Deeper trees include greater design complexity, since more methods and classes are concerned, but the greater the potential for reuse of inherited methods.

Example: Customer is the root and has a DIT of 0. The DIT for Preferred_Customer is 1.

Table 2. Summary Statistics of DIT

Metric	Max. Value	Min. Value	Avg. Value
DIT	4	1	2.5

3. Number of Children (NoC)

The number of children is metric which specify the number of instant subclasses secondary to a class in the hierarchy. The maximum or larger the NoC value that presents it should gives a thought of the potential powerful a parent class has on design. If a class has a huge or larger number of sub-classes, it could need more testing of its methods. The larger or maximum the NoC, the larger the reuse since inheritance is a form of reuse.

Example: Customer has an NoC of 2. NoC for Preferred_Customer is 0 since it is a terminating or leaf node in the tree structure.

Table 3. Summary Statistics of NoC

Metric	Max. Value	Min. Value	Avg. Value
NoC	1	0	0.5

4. Coupling Between Objects (CBO)

CBO metric is a compute of non inherited communications (or interactions) between classes. CBO is calculating or assessing the number of other classes to which a class is coupled. It is calculated or measured by counting the number of distinct non-inheritance related class hierarchies on which a class depends. Extreme coupling is harmful or very dangerous to modular design and it will prevent or prohibit to reuse. The huge no. of independent a class is, it helps the easier to reuse in another application.

Example: Two classes may have extreme coupling i.e. too many messages passing between them. This implies that those classes should be combined into one class.

Table 4. Summary Statistics of CBO

Metric	Max. Value	Min. Value	Avg. Value
CBO	62	0	31

5. Response for a Class (RFC)

RFC is one of the qualities metric that can be defined as the calculation of the set or no. of all methods in class that can be called in reply to a message to an object of the class or by some method in the class. This metric shows or represents at mixture of the complexity of a class through the number of methods and the amount of interaction or communication with other classes. The larger the number of methods that can be called from a class through messages, the greater the complexity of the class.

Example: RFC for Preferred_Customer = 0 (self) + 0 (Customer) + 1 (Credit_Card)= 1

Table 5. Summary Statistics of RFC

Metric	Max. Value	Min. Value	Avg. Value
RFC	55	0	27.5

6. Afferent Coupling(Ca)

Consider the no. of classes within the packages on which the numbers of other packages are depending which specify the package's responsibility. The Afferent Coupling metric finds out the number of classes and interfaces from other packages that depend on classes in the analyzed package. Afferent Coupling is also called as Incoming Dependencies.

Table 6 . Summary Statistics of Ca

Metric	Max. Value	Min. Value	Avg. Value
Ca	1	0	0.5

7. Efferent Coupling (Ce)

The packages comprise no. of classes which are depending on the other packages specify the package's independence. Efferent Coupling (Ce) is also called as Outgoing Dependencies or the Number of Types inside a Package that Depend on Types of other Packages. Ce is a quality metric for packages. In short, this calculate or assess that includes all the types within the source of the measured package referring to the types not in the measured package.

Table 7. Summary Statistics of Ce

Metric	Max. Value	Min. Value	Avg. Value
Ce	2	0	1

8. Coupling Factor (COF)

Coupling Factor (CF) evaluates or determines the coupling between classes. This evaluation can be not including coupling due to inheritance. It is defined as the ratio between the number of actually coupled pairs of classes in a scope (e.g., package) and the possible number of coupled pairs of classes. CF is primarily applicable to object-oriented systems.

Table 8. Summary Statistics of COF

Metric	Max. Value	Min. Value	Avg. Value
COF	62	0	31

9. Data Abstraction Coupling (DAC)

The DAC is metrics which calculates the coupling or pairing complexity caused by Abstract Data Types (ADTs). This metric is concerned with the coupling between classes representing a major aspect of the object oriented design.

Table 9. Summary Statistics of DAC

Metric	Max. Value	Min. Value	Avg. Value
DAC	0	0	0

10. Method Invocation (MIC)

Methods within class are called likely as normal procedures are called. Only they have an object instance identifier i.e. OID prep ended to them. To find out which method is called, it is necessary to know the type of the method.

Table 10. Summary Statistics of MIC

Metric	Max. Value	Min. Value	Avg. Value
MIC	54	0	27

11. Message Passing Coupling (MPC)

This metric assesses or evaluate the numbers of messages passing among objects of the class. A larger or huge number shows increased coupling between analyzed class and other classes in the system. This makes the classes more dependent on each other which increases the overall complexity of the system and makes the class more difficult to change.

Table 11. Summary Statistics of MPC

Metric	Max. Value	Min. Value	Avg. Value
MPC	2	0	1

12. Number of Public Methods (NPM)

This metric measure or count the number of public methods declared in a class.

A greater NPM value can be represent for two various awful conditions in the design of software. First, it can be a indication for a class that is to complex and has too many responsibilities in the analyzed software system. This can be confirmed by looking at Weighted Method per Class metric for the same class. A good example for such a subject is the well known utility class with all those methods, which are used all over the software system.

Secondly, a greater NPM value can be represent for a class that is highly coupled with other parts of the software, because each public method may expose the internally used classes. Or the class acts as some kind of context, to transfer objects between several components. This can be confirmed for possible candidates, by looking at the coupling metrics for the same class, for example the Coupling Between Objects and the Afferent Coupling metric.

Table 12. Summary Statistics of NPM

Metric	Max. Value	Min. Value	Avg. Value
NPM	27	0	13.5

6. COMPARATIVE ANALYSIS

This section describes the comparative study and evaluation to propose the accuracy of proposed tool.

An evaluation methodology which proposed is used for the performance evaluation of existing tools.

Tools	Coupling
Analyst	53.33
VizzAnalyzer	42.33
C&K Java metrics	52
Proposed system	Predicated: Below 40

Table 13 - A Comparison of Performance Evaluation –other tools vs. Proposed System

Tool → **Functionality** ↓	**ANALYSt**	**VIzzANALYZEr**	**C&K JAVA METRICs**	**Proposed system**
Classes	NO	YES	NO	YES
Attributes	NO	YES	NO	YES
Methods	NO	YES	NO	YES
Associations	NO	YES	NO	YES
Multiplicity	NO	YES	NO	YES
Aggregation	NO	NO	NO	YES
Generalization	NO	NO	NO	YES
Instances	NO	NO	NO	YES
Core prototype model	NO	NO	NO	YES
Complete Meta Model	NO	NO	NO	YES
Security Accessibility	NO	NO	NO	YES
Complexity Metrics	YES	YES	YES	YES

Table 14–Functionality support comparison of Proposed System with other tools

7. RESULTS

This section describes the results of proposed tool.

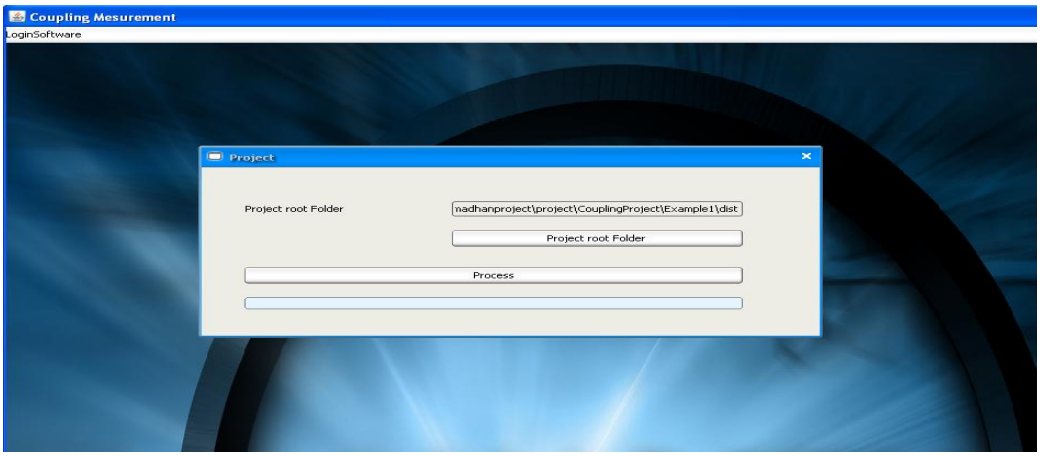

Fig 3. Screenshot Shows the Choosing the Input File From the directory

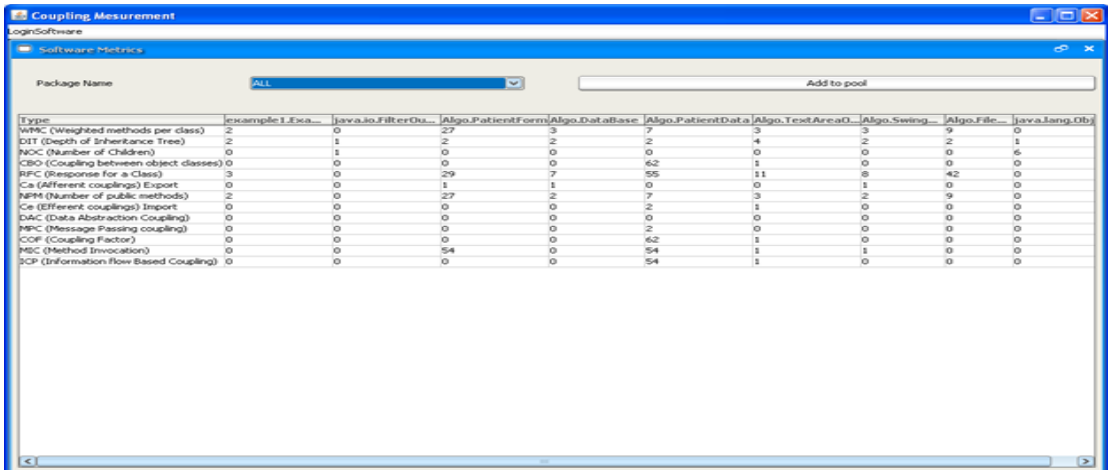

Fig 4. Screenshot shows the Software quality Metric values details for the given input programs.

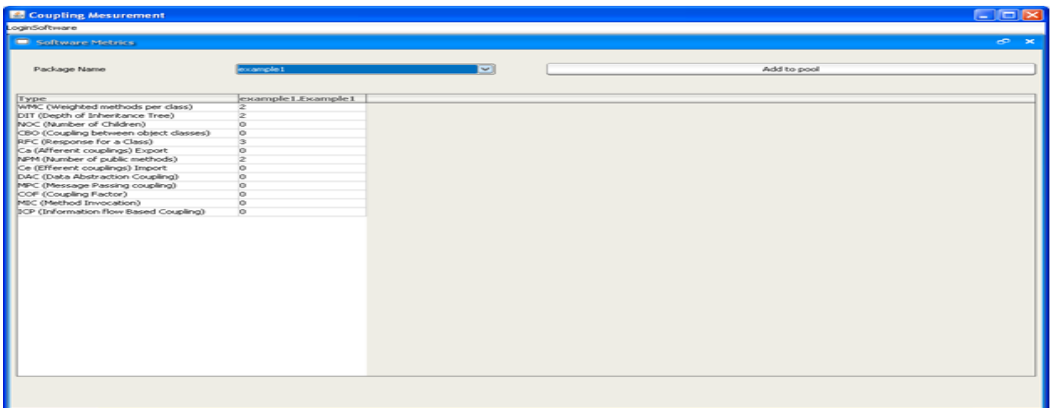

Fig 5. Screenshot shows the Software quality Metric values details for the given input Example.class file programs.

8. CONCLUSION AND FUTURE WORK

The developed system is automated and comprehensive tool which measures product metrics which describe the characteristics of product such as coupling, cohesion etc for Object Oriented program. The proposed work is fully automated eliminating the manual effort required from the developer and analyzer, further because of the elimination of manual work these system is effective, efficient for the reengineering of the software which already in existence with effective utilization of the key resources .

The Automate tool is used to evaluate the quality of the object oriented language application. The tool can be improved to evaluate other Object Oriented languages such as C++, C sharp, etc., Also, it should be enhanced for quality of software system for end user requirement satisfaction, because quality has various standard definition. The improved appraisal tool will be cooperative in assessing the quality in advance to develop awareness for quality issues such as reliability, testability and maintainability.

9. REFERENCES

[1] Ashwin B. Tomar and Dr.Vilas. M. Thakare,"A SYSTEMATIC STUDY OF SOFTWARE QUALITY MODELS", International Journal of Software Engineering & Applications (IJSEA), Vol.2, No.4, October 2011

[2] Ronan Fitzpatrick , Peter Smith and Brendan O'Shea, "Software quality challenges", International Conference on Software Engineering (ICSE 2004)

[3] S.Arun Kumar, T.Arun Kumar and P.Swarnalatha "Significance of Software Metrics to Quantify Design and Code Quality", International Journal of Computer Applications (0975 – 8887), Volume 11– No.9, December 2010

[4] Fernando Brito e Abreu and Walcélio Mel**o, "**Evaluating the Impact of Object-Oriented Design on Software Quality**",** Originally published in Proceedings of the 3rd International Software Metrics Symposium (METRICS'96), IEEE, Berlin, Germany, March 1996.

[5] "Comparing Software Metrics Tools" RudigerLincke, Jonas Lundberg and Welf Lowe Software Technology Group School of Mathematic Mathematics and Systems EngineeringVaxjoUniversity,Sweden{rudiger.lincke|jonas.lundberg|welf.lowe}@vxu.se

[6] A Qualitative Evaluation of a Software Development and Re-Engineering Project Thomas Panas,RudigerLincke, Jonas Lundberg,Welf Lowe Software Technology Group MSI, University of Vaxjo, Sweden

[7] "Beyond Language Independent ObjectOrientedMetrics:Model Independent Metrics" Michele Lanzalanza@iam.unibe.ch Software Composition Group Universit ´a di Berna, Svizzera and St´ephaneDucasseducasse@iam.unibe.ch Software Composition Group Universit ´e de Berne, Suisse

[8] A Qualitative Evaluation of a Software Development and Re-Engineering Project Thomas Panas,RudigerLincke, Jonas Lundberg,Welf Lowe Software Technology Group MSI, University of Vaxjo, Sweden

[9] "Development and Application of Reverse Engineering Measures" in a Re-engineering Tool S. Zhou, H. Yang and P. Luker William C. Chu Department of Computer Science Department of Information Engineering De Montfort University Feng Chia University England Taiwan

[10] "An Empirical Study of Slice-Based Cohesion and Coupling Metrics" Timothy M. Meyers and David Binkley Loyola College in Maryland Baltimore, Maryland 21210-2699, USA {tmeyers,binkley}@cs.loyola.edu

[11] Alshammari, Bandar and Fidge, Colin J. and Corney, Diane (2009) "Security metrics for object-oriented class designs". In: QSIC 2009 Proceedings of : Ninth International Conference on Quality Software , August 24-25, 2009, Jeju, Korea. (In Press)

[12] "New Conceptual Coupling and Cohesion Metrics for Object-Oriented Systems"BélaÚjházi, Rudolf Ferenc, TiborGyimóthy University of Szeged, Hungary Department of Software Engineering ujhazi.bela@stud.u-szeged.hu, {ferenc, gyimi}@inf.u-szeged.hu and Denys Poshyvanyk The College of William and Mary, USA Computer Science Department denys@cs.wm.edu

[13] "Reverse Engineering Component Models for Quality Predictions" Steffen Becker, Michael Hauck, and MirceaTrifu FZI Research Center Software Engineering Karlsruhe, Germany Klaus Krogmann Karlsruhe Institute of Technolgy Software Design and Quality Karlsruhe, Germany Jan Kofro˘n Charles University in Prague Distributed Systems Research Group Prague, Czech Republic

[14] "An Exchange Model for Reengineering Tools" Sander Tichelaar and Serge Demeyer, Software Composition Group, University of Berne, Switzerland, {demeyer,tichel}@iam.unibe.ch

[15] "A Visual Analysis and Design Tool for Planning Software Reengineerings" Martin Beck, Jonas Tr ¨umper and J¨urgenD¨ollner {martin.beck}, {jonas.truemper}, {juergen.doellner}@hpi.uni-potsdam.deHasso-Plattner-Institute – University of Potsdam, Germany