StuConOS 2013

THE SECOND STUDENT CONFERENCE ON OPTIMISATION OF SOFTWARE
25-26 JUNE 2013, UNIVERSITY COLLEGE LONDON, UK

The 2nd Student Conference on Optimisation of Software

25-26 June 2013
University College London
London, UK

EPSRC
Engineering and Physical Sciences Research Council

CREST

Preface

The **StuConOS** 2013 conference is aimed at current undergraduate and master level students in the UK. The conference will give students a chance to submit their work and to have this reviewed by a programme committee of experts. We interpret the word "optimisation" in its very broad sense: anything that improves (in any way) software or the process of making it. It gives students an initial taste of research work and a chance to publish and discuss their ideas with experts.

The programme committee reviewed the submissions and decided on those to be accepted and the prizes to be awarded. All authors received feedback on their work and those accepted for the conference are published in this printed proceedings & are online available and are presented at the conference.

There are prizes for the best papers submitted in the form of a gold and silver medal which will be accompanied by a gift of £300 and £200 respectively.

Conference Organisation

General Chair

Mark Harman	University College London, UK

Programme Chair

Yuanyuan Zhang	University College London, UK

Industrial Chair

Dean Mohamedally	University College London, UK

Local Arrangements Chair

Lena Hierl	University College London, UK

Program Committee

Kelly Androutsopoulos	University College London, UK
Giuliano Antoniol	Ecole Polytechnique de Montréal, Canada
Andrea Arcuri	Simula, Norway
Paul Baker	Visa, UK

Edmund Burke	University of Stirling, UK
Francisco Chicano	University of Málaga, Spain
David Clark	University College London, UK
John A Clark	University of York, UK
Myra B Cohen	University of Nebraska - Lincoln, USA
Massimiliano Di Penta	University of Sannio, Italy
Nicolas Gold	University College London, UK
Wolfgang Grieskamp	Google
Rob Hierons	Brunel University, UK
Jens Krinke	University College London, UK
Kiran Lakhotia	University College London, UK
Emmanuel Letier	University College London, UK
Leandro L Minku	University of Birmingham, UK
Robert Nilsson	Google, Switzerland
Mel Ó Cinnéide	University College Dublin, Ireland
Justyna Petke	University College London, UK
Simon Poulding	University of York, UK
Brian P Robinson	ABB, USA
Federica Sarro	University College London, UK
Saurabh Sinha	IBM, India
Jerffeson Teixeira de Souza	State University of Ceará, Brazil
Nikolai Tillmann	Microsoft, USA
David R White	University of Glasgow, UK
Junchao Xiao	Chinese Academy of Sciences, China
Shin Yoo	University College London, UK

Sponsors

DAASE Project

http://daase.cs.ucl.ac.uk/

Keynote Speaker

Nikolai Tillmann – Microsoft, USA

Nikolai's main areas of research are program authoring on mobile devices, program analysis, testing, optimization, and verification.

He started the TouchDevelop project, which enables end-users to write programs for mobile devices on mobile devices. This project brings the excitement of the first programmable computers to mobile devices such as smartphones.

Nikolai is leading the Pex project, in which he develop together with Peli de Halleux a framework for runtime verification and automatic test case generation for .NET applications based on parameterized unit testing and dynamic symbolic execution. Try out Pex on the web: www.pexforfun.com

He is also involved in the Spur project, where he is working on a tracing Just-In-Time compiler for .NET and JavaScript code.

Previously Nikolai worked on AsmL, an executable modeling language that comes with a compiler and a test generation tool, and the Spec Explorer 2004 model-based testing tool. Together with Wolfgang Grieskamp he developed XRT, a concrete/symbolic state exploration engine and software model-checker for .NET code. Spec Explorer 2007 is based on this engine, which is now productized internally by the Protocol Engineering Team at Microsoft to facilitate quality assurance of protocol documentation (link).

Before coming to Microsoft Research, Nikolai was involved in the development of a school management system in Germany.

UCL

About UCL

UCL was established in 1826 to open up education in England for the first time to students of any race, class or religion. UCL was also the first university to welcome female students on equal terms with men.

Academic excellence and conducting research that addresses real-world problems inform our ethos to this day.

Our research

In the most recent Research Assessment Exercise, 62% of UCL's submissions were ranked at the highest grades of 4* ("of world-leading quality"), or 3* ("internationally excellent"), placing it third in the UK.

UCL attracts the third highest number of academic citations in the UK showing the high esteem and relevance of the institution's research.

Local and global impact

In 2011–2012, UCL ran more than 40 schemes to support start-ups and growing businesses, and helped strengthen more than 300 small businesses in London.

We share our resources and engage with the local community to enrich London's social, cultural and academic life.

CREST Centre

The CREST centre at UCL builds on the three foundations of Program Dependence, Information Theory and Optimisation Algorithms.

On these three foundations we develop ways to analyse, understand and improve software, with applications throughout the spectrum of software development activities. We are widely known for our work on Empirical Software Engineering, Evolutionary Computation, Code Provenance, Quantified Information Flow, Security, Software Testing, Program Slicing and Search Based Software Engineering.

Like other centres at UCL, the CREST Centre is truly multidisciplinary; we apply our algorithms and methods to the analysis and improvement of Digital Humanities and the Arts from which our work of Software Engineering also draws inspiration.

The centre also hosts a series of monthly open workshops.

DAASE Project

DAASE (Dynamic Adaptive Automated Software Engineering) is a four site project between UCL, Birmingham, Stirling and York. The lead at each site is, respectively, Professors Harman, Yao, Burke and Clark, with Professor Harman as the overall project director. The project also has a growing list of industrial partners, which currently includes Air France - KLM, Berner and Mattner, BT Laboratories, Dstl, Ericsson, GCHQ, Honda Research Institute Europe, IBM, Microsoft Research and VISA UK.

DAASE builds on two successful longer larger projects, funded by the EPSRC and which were widely regarded as highly successful and ground breaking. The project also draws inspiration and support from and feeds into the rapidly growing worldwide Search Based Software Engineering (SBSE) community. A repository of SBSE papers and people can be found here.

Current software development processes are expensive, laborious and error prone. They achieve adaptivity at only a glacial pace, largely through enormous human effort, forcing highly skilled engineers to waste significant time adapting many tedious implementation details. Often, the resulting software is equally inflexible, forcing users to also rely on their innate human adaptivity to find "workarounds". Yet software is one of the most inherently flexible engineering materials with which we have

worked, DAASE seeks to use computational search as an overall approach to achieve the software's full potential for flexibility and adaptivity. In so-doing we will be creating new ways to develop and deploy software. This is the new approach to software engineering DAASE seeks to create. It places computational search at the heart of the processes and products it creates and embeds adaptivity into both. DAASE will also create an array of new processes, methods, techniques and tools for a new kind of software engineering, radically transforming the theory and practice of software engineering.

Programme

25th June:

11:30 Arrival and lunch

12:30 Welcome and Introduction by Professor Mark Harman

12:45 Keynote: Nikolai Tillmann, Microsoft Research

13:30 Questions

13:45 Michaela Newell, University of the West of England – StuConOS paper

14:15 Discussion

14:30 BSc Final Year Computer Science Project reports

Anthony Stewart, Durham University

Krishna Patel, Brunel University

15:00 Discussion

15:15 Refreshments

15:45 BSc Final Year Computer Science Project reports

Kim Barrett, University of Warwick

Matthew Smith, University of Warwick

Christiana Agapiou, University of Surrey

James Marchant, University of Warwick

16:45 Discussion

17:00 Wrap-up

17:15 Leave for reception at the London Eye

19:00 Reception at the London Eye

19:30 Leave for dinner at China Town

20:00 Dinner

26th June:

10:00 Arrival, coffee, tea and pastries

10:30 Pavlo Bazilinskyy & Markus Brunner, University of St Andrews – StuConOS paper

11:00 Discussion

11:15 BSc Final Year Computer Science Project reports

Karol Pogonowski, University of Edinburgh

Christy Kin-Cleaves, Durham University

Stephen McGruer, University of Edinburgh

Deepak Ramchandani Vensi

12:15 Discussion

12:30 Sandwich lunch at the venue

13:30 BSc Final Year Computer Science Project reports

Horatio Caine, University of Birmingham

Razvan Ranca, University of Edinburgh

Jibran Khan, University of Edinburgh

Barney Jackson, University of Edinburgh

14:30 Discussion

14:45 Refreshments

15:15 BSc Final Year Computer Science Project reports

Aaron Cosgrove, Manchester Metropolitan University

Arun Kumar, University College London

Mariyana Koleva, University of Edinburgh

16:00 Discussion

16:30 Wrap up and Prize Announcement

17:00 Finish

Shortlist for the BSc Project Prize

- Presenting their work –

Christiana Agapiou. Analysis and monitoring of tremor - University of Surrey

Kim Barrett. Mario and AI - University of Warwick

Horatio Caine. Prediction and Classification for Optimisation of Object Tracking in Market-Based Networks of Distributed Smart Cameras - University of Birmingham

Aaron Cosgrove. Fuzzy Decision Trees for Medical and Financial Applications - Manchester Metropolitan University

Barnaby Jackson. The effectiveness of previews on search engine results - University of Edinburgh

Jibran Khan and Michael Rovatsos. Plan Recognition in RISK - University of Edinburgh

Christy Kin-Cleaves. Backgammon with Variable Luck - Durham University

Mariyana Koleva. Automated Classification of Butterfly Images - University of Edinburgh

Arun Kumar. An English Writing Search Engine – UCL

James Marchant. The Effect of Placement Strategy on Convention Emergence - University of Warwick

Stephen McGruer. Automated Compiler Optimization: Does it really work? - University of Edinburgh

Krishna Patel. Assessing the Feasibility and Desirability of a Genetic Algorithm Based Routing Protocol - Brunel University

Karol Pogonowski. Cloud computing on Heterogeneous cores: MapReduce on GPUs - University of Edinburgh

Deepak Ramchandani Vensi. Groupee - A solution for Group Social Innovation - University of Bath

Razvan Ranca. Reconstructing Shredded Documents - University of Edinburgh

Anthony Stewart. A study of Ant Colony Algorithms and a potential application in Graph Drawing - Durham University

Matthew Smith. Giving Up Smoking: Modelling how Social Networks Impact upon the Breaking of Habits - University of Warwick

BSc Project Prizes

The Effect of Placement Strategy on Convention Emergence

James Marchant - University of Warwick

Automated Classification of Butterfly Images

Mariyana Koleva - University of Edinburgh

StuConOS Best Paper Awards

Gold Prize

Improving Testing of Complex Software Models through Evolutionary Test Generation

Michaela Newell - University of the West of England

Silver Prize

Performance Engineering and Testing: The Challenges on Mobile Platforms

Markus Brunner and Pavlo Bazilinskyy - University of St Andrews

Improving Testing of Complex Software Models through Evolutionary Test Generation

Michaela Newell
Department of Computer Science and Creative Technologies
University of the West of England
Bristol BS16 1QY United Kingdom
michaela2.newell@live.uwe.ac.uk

ABSTRACT
Considerable cognitive effort is required to write test cases for complex software and fix any defects found. As the generation of test cases using evolutionary computation has a long established track record, this paper explores whether this pedigree can be exploited to improve efficiencies in larger testing suites that typically address complex software models. A genetic algorithm has been designed and implemented with complex software models in mind, and then trialled against five real world programs that vary in scale and complexity. Results show that test case generation using an evolutionary algorithm on average can improve the number of coverage goals met by 75.83%. Therefore we conclude that even with complex models that have thousands of objects improvements can be made.

Categories and Subject Descriptors
D.2.5 [**Software Engineering**]: Testing and Debugging – *Testing tools (e.g., data generators, coverage testing).*

I.2.8 [**Computing Methodologies**]: Problem Solving, Control Methods, and Search – *Heuristic methods.*

General Terms
Algorithms, Design, Human Factors

Keywords
Automatic Test Generation, Evolutionary Algorithm, Metaheuristic Search, Unified Modelling Language, Complex Data Models

1. INTRODUCTION
Manual software testing of complex software is a cognitively demanding task and so can be expensive, time consuming and occasionally unreliable [1]. Figures released for the USA suggest that approximately $20 billion each year could be saved if more efficient and effective software testing was performed prior to deployment and release. [2]. The need to improve on this situation is significant as in addition to the loss of considerable amount of money, failures of complex safety-critical software systems potentially put human life in jeopardy. As a spectacular example of a complex software systems failure, the European Space Agency estimates losses of $500,000,000 caused by the launch failure of the Arianne 5 rocket in 1996; the root cause of the failure is thought to be inadequate testing coverage [3].

Evolutionary Test Generation (ETG) has attracted significant research interest and shows great promise in reducing the development costs and improving the quality (and hence the level on confidence) in the software under test [4]. Within the field of Search Based Software Engineering (SBSE) [5], many meta-heuristic search techniques have been applied which treat the generation of test cases as a search problem. Meta-heuristic search approaches encode candidate solutions using a problem specific representation, and fitness functions and operations to preserve solution diversity. The technique is typically measured on how expensive, effective and scalable the algorithm is at reaching the test objectives [6]. A widely applied objective fitness function used in meta-heuristic search for test cases is branch coverage [7], where the goal is to arrive at a restricted number of test cases that achieve the maximum degree of branch coverage. However, generating a set of test cases for software systems of realistic complexity presents a challenge not only due to the size of the search space expanding rapidly, but furthermore, many researchers acknowledge solving a complex search problem means there is no optimal or exact solution [8]. This paper therefore sets out to design and implement a genetic algorithm that exploits models of complex software, specifically object-oriented class models, as a basis for generating test cases.

2. BACKGROUND
Evolutionary Algorithms (EA) typically use a population of individuals, rather than one individual candidate. The algorithm then uses optimisation techniques inspired by the biological evolution processes; reproduction, mutation and selection. EAs consist of a number of varying techniques including: genetic algorithms, genetic programming and evolutionary programming. Genetic Algorithms (GA) are arguably the most well known form of EA [9]. Genetic algorithms require three components in order to achieve effective search: a solution representation, a measure of solution fitness and a mechanism for diversity preservation. A representation can typically take the form of real numbers, binary digits or floating point numbers. Examples of GAs using more complex data structures have attempted to address some challenging problems such as scalability, predictability and robustness [10].

There are many methods of generating test cases, including search based [8][10], model based [11][12][13][14] and specification based [15]. Model based test generation is very different to the search techniques discussed. The tests are generated from modelling languages including the most widely used [11], Unified Modelling Language (UML) [16]. The benefit of this method is that in many cases, designs in the form of UML have already been completed; consequently less additional effort is required. Data can be gathered from various UML diagrams including: use cases [14], interaction diagrams [15] or a combination of diagrams [16].

Figure 1 summarises the different techniques that can be used and the level of testing that they are suitable for:

	Unit Testing	Integration Testing	Functional Testing
Search Based Technique		McMinn [8] Harman [10]	McMinn [8] Harman [10]
Model Based Technique	Prasanna *et al.* [11] Nebut *et al.* [12] Swain *et al.* [14]	Prasanna *et al.* [11] Tonella and Potrich. [13] Swain *et al.* [14]	Swain *et al.* [14]
Specification Based Technique			Liu and Nakajima [15]

Figure 1. Techniques suitability to different stages of testing.

Figure 1 summarises all of the previously mentioned frameworks by their technique and how the authors assess their suitability in the various stages of software testing. The figure shows that the only framework that is suitable for all levels is a model based technique proposed by Swain *et al.* [16]. Additionally Figure 1 highlights that various techniques and frameworks can be adapted, one most appropriate to the problem.

3. PROPOSED APPROACH

The proposed approach uses a combination of a model and search based technique to generate test cases and to address the challenges of a complex data model. Other authors have also proposed combining these two techniques and their methods show promising results [17]. A description of how this approach is implemented is included in the following sections.

3.1 Problem Encoding

A prerequisite of the system is that the user must input a UML Class Diagram. As the tool is an initial prototype only one structure of UML is currently supported. The structure supported is the automatically generated structure of StarUML [18]. An example of the structure can be seen from Figure 2.

```
<XPD:OBJ name="OwnedElements[0]" type="UMLClass"
guid="95FFK+ln1kOVRD4GJYRJhQAA">
<XPD:ATTR name="Name" type="string">Class1</XPD:ATTR>
<XPD:REF name="Namespace">YMt6yt/Y90qiBdvoCPhIcwAA</XPD:REF>
<XPD:ATTR name="#Views" type="integer">4</XPD:ATTR>
<XPD:REF name="Views[0]">0tQXfFFEhkaabBimWRCnpAAA</XPD:REF>
<XPD:REF name="Views[1]">Phczm+38hEqdvBhZpBDf3AAA</XPD:REF>
<XPD:REF name="Views[2]">s/+vlwh5EEC9dhihI0tb5gAA</XPD:REF>
<XPD:REF name="Views[3]">zYgdJGmGL0uBTrgKDz2F3wAA</XPD:REF>
<XPD:ATTR name="#Operations" type="integer">2</XPD:ATTR>
<XPD:OBJ name="Operations[0]" type="UMLOperation"
guid="zOXxGE8kH0CZy0upBp5RnAAA">
<XPD:ATTR name="Name" type="string">method1</XPD:ATTR>
<XPD:REF name="Owner">95FFK+ln1kOVRD4GJYRJhQAA</XPD:REF>
</XPD:OBJ>
<XPD:OBJ name="Attributes[0]" type="UMLAttribute"
guid="hPKc2Pm4k0arRtSG5BMetgAA">
<XPD:ATTR name="Name" type="string">var1</XPD:ATTR>
<XPD:ATTR name="Visibility"
type="UMLVisibilityKind">vkPrivate</XPD:ATTR>
<XPD:REF name="Owner">95FFK+ln1kOVRD4GJYRJhQAA</XPD:REF>
</XPD:OBJ>
</XPD:OBJ>
```

Figure 2. StarUML structure: class, attribute and operation.

However, not all models in the case study are as simple as the one that can be seen in Figure 2. One of the models used to validate how the program manages complexity can be seen in Figure 3.

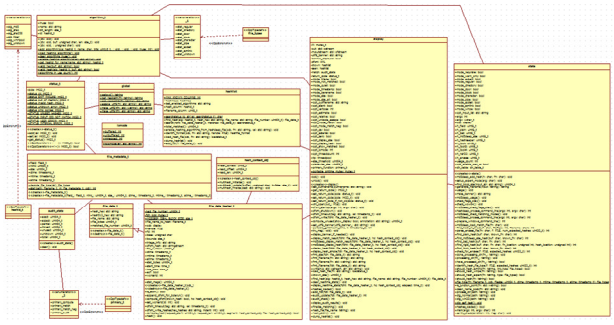

Figure 3. Class diagram for md5deep and hashdeep.

Figure 3 illustrates a diagram used to check the frameworks ability to handle model complexity, with 18 classes and hundreds of attributes and operations. The framework begins by parsing the UML to gain information. In order to identify relevant information within the UML, the required information must first be identified by its XML tag. The tag '<XPD:OBJ>' will differentiate between the type of object (Class, Variable or Method). Inside the objects the parsing looks for the tag '<XPD:ATTR>', which will identify information relating to the object such as the object name.

3.2 Solution Representation

The tool splits by each object narrowing the search space. There is then a separate method that gets specific information from the object. For example, the method can be called to return all method names. Both of these methods are generic to allow for future expansion to obtain more information from the UML.

There are four methods that request information from the generic methods. These four methods are split into types of objects. The logic of these methods is: for each class diagram check for all classes and for each class check for all variables and methods. Currently these four classes return the object names and in the structure specified, add them to a vector. This initial structure includes all the objects and is used later in the fitness function. The benefit of obtaining this information from a design diagram as opposed to the program's code is that we are assuming the design diagrams are correct and we cannot assume this for the program. An example of an initial vector can be seen in Figure 4.

[ClassDiagram1, Class1, var1, var2, method1, method2, Class2, var3, var4, method3, method4]

Figure 4. Example of an initial vector.

Figure 4 shows a genotype representation, Figure 5 the phenotype.

[ClassDiagram1, Class1, method1, Class2, var3, var4]

Figure 5. Example of an individual.

3.3 Diversity Preservation

The standard selection schemes include: tournament, ranking and proportional truncation selection [19]. This framework uses three techniques: tournament selection, one point crossover and single point mutation. The tournaments run by pairing each individual in the population together at random. The winner of the tournament is determined using a fitness proportional selection, as opposed to the absolute fitness value that is determined at the end of each generation. The winner of each tournament is selected for crossover. One point crossover is where one point is selected at

random in the individual and all the data beyond that point is swapped between the two parents, resulting in offspring.

3.4 Fitness Operation

The fitness function assesses the population for coverage goals. Each individual in the population is assessed for its' coverage. A higher coverage can be achieved by including the testing of each object. The individual's assessments are then used for diversity preservation. Once complete, the population's fitness is assessed. The population can increase its fitness score by increasing the number of objects that will be tested. The score is awarded by giving one point if an individual contains an element that exists in the initial vector.

3.5 Genetic Algorithm

The general scheme in psuedocode [20] can be seen in Figure 6.

BEGIN

INITALISE population with random candidate solutions;

EVALUATE population;

REPEAT UNTIL (TERMINATION CONDITION is satisfied) DO

1. *SELECT* two candidates at random;
2. *EVALUATE* each candidate;
3. *SELECT* two candidates at random;
4. *SELECT* crossover point for each candidate;
5. *SELECT* parent for each candidate;
6. *SELECT* child for each candidate;
7. *RECOMBINE* parent of candidate with child of other candidate;
8. *SELECT* two candidates at random;
9. *EVALUATE* each candidate;
10. *SELECT* highest candidate for the next generation;
11. *MUTATE* the resulting offspring;

OD

END

Figure 6. Scheme of an evolutionary algorithm in psuedocode.

4. EXPERIMENTAL METHODOLOGY

Each test will be structured in the same way in order to guarantee the fairness of a comparison. Each test will have a main variable, the program. The test vehicles are five real world programs. Each program varies in the number of objects and the complexity of the model.

The algorithm parameters are: 20 runs for each program, similar to the works of Forrest *et al.* [21]. The number of generations will be 100, similar to the works of Arcuri *et al.* [22]. The population size will be 40 similar to sizes in previous literature [21][22]. 100% of individuals are paired together in a tournament selection which is a common practise in GA's [20]. The crossover rate is 75% which is considered to be ideal [23]. The mutation rate is 5% as this typically shows the best performance [23].

5. RESULTS

The fitness curve for Program 2 is shown in Figure 7.

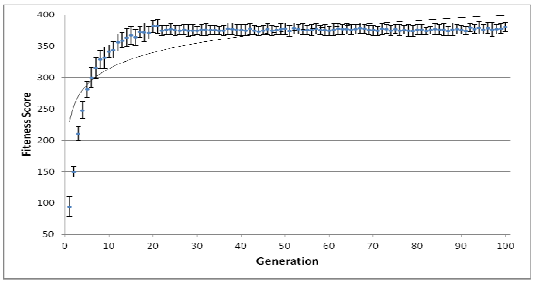

Figure 7. Mean fitness score for 20 runs of Program 2 over 100 generations on a logarithmic scale.

As it can be seen in Figure 7, the fitness score peaks before the 20th generation, which is why every further run used 20 generations, instead of 100. The highest standard deviation point is 17, which is observed at the 7th and 9th generations. As it can be seen, the later generations have a lower standard deviation this is expected as at the beginning the diversity between individuals is larger due to the size of the exploration space is larger. As the candidate solution begins to form the search space decreases, lowering the deviation.

In order to test the results by model complexity and the frameworks ability to scale, Figure 8 shows the total number of objects for each program used in testing. The number of objects directly influences the complexity of the model.

Program Number	Program Name	Length (Total Number of Possible Objects)
1	Bouncy Castles – Open PGP	1478
2	Autopsy – Keyword Search	15
3	md5deep and hashdeep	1872
4	pmd	117
5	TrueCrypt	504

Figure 8. The total number of objects per program.

All of the programs chosen were open source, written in either Java or C++. Source code, test cases and design diagrams were freely available. However, the source code is not used by the framework. The design diagrams are used in the test generation which is then compared to the manual test cases to assess improvement. Figure 9 shows the fitness curve relative to the total number of possible objects listed in Figure 8.

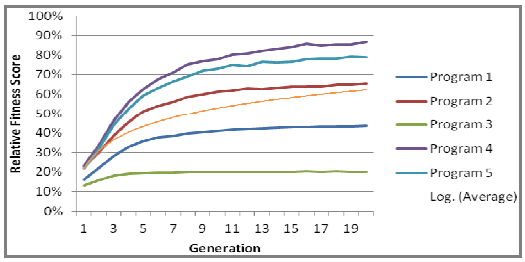

Figure 9. Fitness scores relative to the total number of objects.

Figure 9 shows that every program has a similar fitness curve. The small decreases in the fitness curve are due to the program using a generational model, as opposed to steady state. Figure 9 also shows that the scalability affects the curve. Program 3 has the highest number of objects and the lowest relative fitness score. Program 4 has the second lowest number of objects and the highest fitness score. This indicates that the programs complexity

affects the fitness score. The percentage improvement when comparing automatically generated test cases to manual test development shows that there is a larger room for improvement when the programs complexity is greater. For example, Program 3 achieved a percentage improvement of 95% whilst; Program 1 achieved an improvement of 77%. This could arguably be due to the difficulty of developing tests cases manually for larger programs.

6. CONCLUSION

Results show that significant improvement can be made when using automatic test generation as opposed to manual development. The improvement can be made on software that already exists and is used in the public domain. The complexity affects the final fitness and based on the limited testing, complex programs with more objects, score slightly lower fitness scores. In order to achieve a reasonable relative score (50% or higher) the program size has to be around 1000 objects or lower. However, using the final code base, the average improvement of the programs used was 75.83%. This is strong evidence to suggest that the proposed approach addresses the challenges of a complex model.

Limitations include: the user must have an accurate class diagram written in StarUML. However if desired the 'UMLParser' can be modified to accept various structures chosen by the UML tools. Another limitation is that class diagrams do not typically contain information such as boundary conditions and run time information. The class diagram is currently used as a pre validation to make sure all the objects in design are included. Future work would include obtaining information such as run time information from a sequence diagram or other suitable method. Currently boundary conditions are taken from the user via a GUI, in order to achieve complete automation; this to could be taken from a UML diagram or alternative method. Lastly a multi-objective fitness function would be preferable. After viewing the test cases generated, whilst they make a significant improvement in terms of object coverage, it would be beneficial to improve test cases based on variables such as: length of test, diversity in test set, execution time *et cetera*.

The challenges of a complex model have been addressed. Each of the five programs tested against, were variant in complexity. The improvement in efficiency appears not to be based on the size of the program and/or testing suite but on the quality of the current tests. Reinforcing that irrespective of the problem model complexity, automatic test case generation can be an improvement on manual test development.

7. ACKNOWLEDGMENTS

Thanks to Dr Chris Simons for his continued support.

8. REFERENCES

[1] Katanić, N., Nenadić, T., Dečić, S., Skorin-Kapov, L. 2010. *Automated generation of TTCN-3 test scripts for SIP-based calls.* MIPRO, 33rd International Convention.

[2] Tassay, G. 2002. *The Economic Impacts of Inadequate Infrastructure for Software Testing.* Final Report.

[3] Kosindrdecha, N. and Daengdej, J. 2010. *A Test Case Generation Process and Technique.* Journal of Software Engineering. 265-287.

[4] Clark, J., Mander, K., Mcdermid, J., Tracey, N. 2002. *A Search Based Automation Test-Data Generation Framework for Safety-Critical Systems*. 1-41.

[5] Harman, M. 2010. *Why the Virtual Nature of Software Makes it Ideal for Search Based Optimization.*

[6] Shaukat, A., Lionel C. B., Hadi, H., Rajwinder K. 2010. *A Systematic Review of the Application and Empirical Investigation of Search-Based Test Case Generation.* IEEE Transactions on Software Engineering. 742-762.

[7] Fraser, G., Arcuri, A. 2011. *Whole Test Suite Generation.* Software Engineering, IEEE Transactions.

[8] Pavlov, Y., Fraser, G. 2012. *Semi-automatic Search-Based Test Generation.* IEEE Fifth International Conference on Software Testing, Verification and Validation.

[9] McMinn, P. 2004. *Search-based Software Test Data Generation: A Survey.* 105-156.

[10] Harman, M. 2007. *The Current State and Future of Search Based Software Engineering.* Future of Software Engineering, 2007. 342-357.

[11] Prasanna, M., Sivanandam, S., Sundarrajan, R., Venkatesan, R. 2005. *A survey on Automatic Test Case Generation*. Academic Open Internet Journal.

[12] Nebut, C., Fleurey, F., Le Traon, Y., Jezequel, J. 2006. *Automatic test generation: a use case driven approach.* IEEE Software Engineering. 140-155.

[13] Tonella, P., Potrich, A. 2003. *Reverse Engineering of the Interaction Diagrams from C++ Code.* IEEE International Conference on Software Maintenance, 159-168.

[14] Swain, A. K., Mohapatra, D. P., Mall, R. 2010. *Test Case Generation Based on Use case and Sequence Diagram.* Int. J. of Software Engineering. 21-52.

[15] Liu, S., Nakajima, S. 2010. *A Decompositional Approach to Automatic Test Case Generation Based on Formal Specifications.* Fourth International Conference on Secure Software Integration and Reliability Improvement. 147-155.

[16] Object Management Group. 2013. http://www.uml.org/ [2 June, 2013].

[17] Neto, A., de Freitas Rodrigues, R., Travassos, G. 2011. *Porantim-Opt: Optimizing the Combined Selection of Model-Based Testing Techniques.* ICSTW. 174-183.

[18] StarUML. 2005. http://staruml.sourceforge.net/en/ [26 January, 2012].

[19] Legg, S., Hutter, M., Kumar, A. 2004. *Tournament versus fitness uniform selection.* 2144-2151.

[20] Eiben, A.E., Smith, J.E. 2003. *Introduction to Evolutionary Computing.* 2nd edn.

[21] Forrest, S., Nguyen, T., Le Goues, C., Weimer, W. 2009. *A Genetic Programming Approach to Automated Software Repair*. 947-954.

[22] Arcuri. A., Yao, X. 2007. *Coevolving Programs and Unit Tests from their Specification*.

[23] Andrade, V.A., Errico, L., Aquino, A.L.L., Assis, L.P., Barbosa, C.H.N.R. 2008. *Analysis of Selection and Crossover Methods used by Genetic Algorithm-based Heuristic to solve the LSP Allocation.*

Performance Engineering and Testing

The Challenges on Mobile Platforms

Pavlo Bazilinskyy
University of St Andrews
School of Computer Science
pb52@st-andrews.ac.uk

Markus Brunner
University of St Andrews
School of Computer Science
mb246@st-andrews.ac.uk

ABSTRACT
This paper discusses the challenges which mobile devices raised in terms of Performance Engineering and Perfomance Testing. Possible improvements and adaptions to increase the awareness of importance of both areas are presented.

Categories and Subject Descriptors
D.2.8 [**Software Engineering**]: Metrics

Keywords
Software Optimisation, Perfomance Engineering, Performance Testing, Software Metrics

1. INTRODUCTION
Performance is a universal quality of software that is affected by all aspects of the code, design and means of execution. A recent study reports that at least half of executives of IT companies have encountered performance issues in as many as 20% of their projects [1]. Acceptance of software products and how successful they are depends largely on experience that users of the system receive from using the applications and poor performance is often linked with dissatisfaction and frustration of clients. Ensuring high performance of software systems is especially critical in mobile devices.

Adoption of mobile devices is the fastest technology adoption curve in history. Mobile application market is expected to reach $25 billion by year 2015 [10]. Success of applications deployed to mobile devices depends on their responsiveness: in case of poor performance of the application levels of revenue, brand and customer loyalty decline [8]. Moreover, as reported by IBM, 66% of mobile device users are less likely to purchase goods or services from a company following poor performance of the company's mobile application [7].

Clearly, performance is crucial for software solutions targeted to mobile applications market. This paper focuses on Performance Engineering as a whole and on its practical application during the development process - Performance Testing. The focus of this work is on how Performance Engineering and Testing can be organized and optimised on mobile platforms whose advent over the course of the recent years entailed new challenges for software manufacturers.

2. BACKGROUND
2.1 Performance Engineering
Software Development for mobile platforms is one of the the fastest growing fields of Software Engineering. The connection between business recognition and application success when applied to mobile-based software is now becoming apparent [17]. User requirements for software performance are often neglected before it is too late and required changes become costly [13]. Similarly to other fields of Software Engineering, Performance Engineering is limited by intense schedules, badly designed requirements, and over-optimistic objectives [19]. However, these days it is becoming common knowledge that all types of software have specific performance requirements that need to be taken into consideration on the stage of conceptualizing.

Working on undesired changes and going over budget may be avoided by following principles of Software Performance Engineering or just Performance Engineering, which is focused on software system performance and scalability [13]. The objective of this field of Computer Science is to meet responsiveness goals through modeling software requirements and variations of design. Received models are then used to measure and evaluate the expected performance. Evaluation is done by estimating trade-offs in functions, size of hardware, accuracy of obtained results, and resource requirements. If the results of testing do not meet the requirements, the process is repeated with different models [15]. This series of actions starts on the design stage, but it also continues during the implementation stage of the project. Performance Engineering ensures that more accurate models of software and its performance estimates may be developed.

Performance Engineering applied to software development normally includes all of the following activities [19]:

Identify concerns : the qualitative evaluation of influences of performance goals.

Define and analyse requirements : by using UML or special scenario languages outline the operational profile, estimate workload, delay and throughput require-

ments, and scenarios describing what the system is expected to output.

Predict performance : by utilising scenarios, architectures, design outlines estimate expected performance by modeling the interaction of the behaviour with the resources.

Performance Testing : check performance of the system under normal and stress conditions. This paper focuses on this aspect of Performance Engineering.

Maintenance and Evolution : estimate the effect of potential changes and additions, such as added features, migration to a new platform, migrating to new web application platforms.

Total System Analysis : plan how software will behave in the deployed system.

Further, the latest report of Eurostat indicates that most countries in Europe have more mobile subscriptions than inhabitants [2]. Our generation is experiencing the evolution of the mobile platform: to succeed we need to constantly rethink how we comprehend ways in which customers and employees communicate with us, and universal access to information is in great demand now [4]. However, traditional approaches that are utilised in Software Performance Engineering may no longer be applied to the mobile platform - the rate of progress in research connected to Performance Engineering is much slower than that of evolution of the mobile platforms. Additionally, analysing the software performance of software created for mobile devices is complicated because software architects usually need to find equilibrium between the building blocks of Performance Engineering, performance and scalability, and other quality-of-service attributes such as manageability, interoperability, security, and maintainability [9]. Nevertheless, this is an important research area due to popularity of mobile devices in today's world.

2.2 Performance Testing

As mobile devices are characterised by limited performance capabilities (e.g. memory, battery life, computing power, etc.), Software Optimisation and Performance Engineering are two crucial elements in the software development process. The performance progress of mobile platforms in the recent years was imposing, resulting in even more powerful devices. However, energy is the limiting factor which sets the border for further progression. Application developers must understand the trade-offs between performance and battery life in order to optimise resource utilisation [16]. This fact emphasises the importance of Performance Testing on mobile platforms. The practice shows that a majority of application developers and software houses which are deploying the applications put only limited focus on this aspect. Owners of an early version of Apple's iPhone will probably already have faced the challenge of installing and using a new application for their 2-4 year old phone. This very short life cycle derogates the users' experience and takes the wind out of the sails of the mobile platforms' advance.

Weyuker et. al claimed that reasons for the lack of Performance Testing in order to produce resource-efficient software can often be attributed to "the lack of performance estimates, the failure to have proposed plans for data collection, or the lack of a performance budget" [18]. Furthermore his experience showed that a lack of sufficient planning for performance issues is another precursor for bad software. Even though these findings were determined already 13 years ago, we can project them onto nowadays' mobile app development. One of the reasons for a lack of Performance Testing on mobile platforms is the fact that applications can be developed by third-parties which do not have to fulfill certain quality requirements and standards.

Amongst the countless magnificent app-concept that were developed by third-party developers (e.g. groups of students, programmers, freelancers, start-ups, etc.), there are very few project teams which have taken Performance Testing into serious consideration. Applications which permanently strain the resource pool of a mobile device have a bad influence on the device's energy consumption and are subsequently shortening the battery life. A solution could be to restrict developers with imposing stricter developing guidelines or quality standards on them. However, this approach would lead to a decline of diversity in terms of widely available and easy accessible SDKs (Software Development Kits). Considering that many apps were developed in home offices or dorm rooms, this would lead to a loss of creativity and dynamics in the developer community.

3. RELATED WORK

Literature on Performance Engineering in mobile devices is limited. We think it may be explained by the fact that this area of research is still seen as a cutting-edge field of software engineering and measuring performance of projects developed for mobile platforms is currently a developing area of research. There are virtually no literature sources that address the issue of applying performance engineering to mobile development projects. However, ideas on how to approach the issue could for example be received from Harman an Smith's work, who review the key ingredients required for successful Software Optimisation and describe the benefits that are likely to acrue from growing work in the field, while considering problems and challenges for future work. [5]

In regards of Performance Testing, research has been conducted on the generation of frameworks and performance metrics. However, there is very few literature which deals with delivering these concepts into practice.

4. METHOD

Similarly to the situation with published research papers that touch the issue of maintaining Performance Engineering in mobile devices, a pool of methods that are available for addressing the issue is also limited. After finishing our literature review and realising that the scientific community is not fully aware of the problems of Performance Engineering and Testing with mobile software we decided to focus our writing on outlining existing problems. Additionally, our main concern about methods used to implement performance analysis of the process of engineering mobile applications is that they are mostly created using agile development techniques. Performance Engineering is overly complicated when they are used. Results obtained from analysis of performance may be improved by setting multiple objectives. We propose using

Search Based Software Engineering (SBSE) for investigating what objectives may be defined for a project that deals with mobile devices.

5. IMPROVING THE PERFORMANCE ON MOBILE DEVICES

5.1 Performance Engineering

The work of Smith and Williams points out that Software Performance Testing should be started early in the software development process: as early as at the stage of conceptualising the system [13]. This approach works best with systems that have safety as the main attribute. However, due to limited access to resources mobile software companies tend to focus on the commercial side of their projects and neglect Performance Engineering until problems with their published on the market applications start to arise. One may argue that Performance Engineering should be integrated into mobile software development process, rather than it being a mere addition to the process [14].

Additionally, mobile development projects tend to have short time dedicated to the actual development phase. Agile development is commonly used with mobile development to produce multiple releases that incorporate small changes. Problems with performance in projects often deployed to multiple mobile platforms are difficult to notice. Therefore, Performance Engineering may be an important asset to projects created in this way. The problem of measuring performance of the final mobile software product received through following rapid development is defined by Barber [3]. One may find it hard to add Performance Engineering into the agenda outlined for the project that is created using agile development. This issue may be addressed by defining precise, quantitative performance objectives as explained by Smith and Williams [14]. They help to explicitly outline what is expected from the system and once the milestone is reached, quantitatively determine whether the software meets that objective. Developers may also wish to define more than one objective. During the modeling process yielded by the model results should be compared to what is expected from a particular objective. It helps to realise if the project faces a risk of not meeting the objective. If that is a case, appropriate actions should be taken. As soon as results of performance tests can be obtained, one is able to see whether or not created software meets the objective.

The objectives for the project may be obtained by using SBSE. Its main objectives could be defined as: 1. Choose the representation of the problem. 2. Define the fitness function [5]. The fitness function determines the best solution by using the search algorithm of choice that can differentiate between solutions and quantify achieved progress [6]. SBSE is a relatively young field of science and its revolutionary capabilities may be used to optimise Performance Engineering so that it could be adapted to development of mobile devices.

5.2 Performance Testing

The motivation for Performance Testing is often not existent amongst developers, since the users' experience in regard to performance plays just a minor role in early stages of the software development process. The outcomes of a missing Performance Testing component stay latent as long as the produced applications are deployed to a set of homogeneous devices as there is no difference in performance between them. We have seen this phenomenon during the desktop era, when computers became more and more capable in terms of data processing and memory. The application and hardware lifecycles were aligned to each other - as hardware got old, applications got old and were replaced by newer versions.

The contemporary computing environment is in contrast characterised by a variety of different platforms and technologies which differ in terms of performance, mobility, energy efficiency, form factor, etc. Just a few years ago people used a single desktop computer or laptop in their homes and replaced this device when they thought that the time had come to move on to a better product. This situation has radically changed - these days people use a mix of laptops, mobile phones, desktop computers and tablets for different tasks. These devices feature different characteristics as they are designed to serve different purposes.

These diversified characteristics result in semi-latent performance issues which are highly influencing the users' experience. Semi-latent therefore, because many applications do not consider specific technological capabilities of the target platforms. A data-intensive video application for a mobile phone might run smoothly on the latest hardware but could cause problems on an older version of it. The cause of this problem is enrooted in the evolution of the programming languages and their frameworks which were developed towards a higher level of abstraction in order to be able to ignore the underlying hardware. This development might have made the lives of the developers easier but has its downsides in regard to scalability and resource-efficiency. Schmidt claimed that today's programming concepts provide abstractions of the solution space (i.e. the hardware itself) rather than abstractions of the problem space in terms of application domains (e.g. health care, insurance, biology, etc) [12].

There are two stages which have to be fulfilled in order to integrate Performance Testing into the development process. Firstly, software requirements (i.e. metrics) have to be clearly defined. These metrics are the ground on which Performance Testing can be build upon. The "body of research and practice in developing metrics for traditional software" is rich; however, there has been little research on how these metrics can be related to mobile applications [11]. Once these metrics have been established, it comes to the implementation stage, where the performance of applications has to be evaluated and tested against the defined metrics.

There is a substantial problem which can be detected when it comes to Performance Testing. Firstly, developers test their software on modeled pre-deployment environments, which are just another form of abstraction of the production environment. Moreover, performance is usually tested after the development process rather than being continuously assessed during it. Improvements on that can be achieved by tightening the coupling between Software Development and Performance Testing. According to Thompson, Model-driven Development (MDE) is a promising solution to this problem [16]. The concept of MDE is to develop and test

software in early stages of the design process to identify key characteristics, such as power consumption, memory requirements, etc.

The challenges of Performance Testing are researched and theoretical concepts have been developed. Although there is still much space for improvements and more research, since these concepts have to be delivered to the mobile application developers. Integrated Development Environments (IDEs) still lack of support for Performance Testing, but with further advance of mobile devices, developers will not be able to avoid the adoption of Performance Testing, at least if they want to succeed in the competition. Not only the integration of Performance Testing frameworks and tools in IDEs has to be improved, but the importance of Performance Engineering and Testing has to be propagated in the whole field of Software Engineering.

6. CONCLUSION

We briefly touched a problem of Performance Engineering and Performance Testing in projects that deal with applications intended for mobile devices.

Software used in mobile devices is normally created using agile development techniques and it is difficult to use conventional means of assessing and improving performance due to the nature of frequent updates submitted by the team working on the project. We suggest improving performance of mobile applications by outlining multiple precise, quantitative performance objectives. Results received during the modeling process are then compared to what a particular objective requires. This process can indicate that the project is not likely to meet the objective. If that is true, appropriate actions can be taken. Objectives defined for projects may be created using Search Based Software Engineering techniques.

We assessed that Performance Testing is not consistently integrated in the software development cycle yet. To emphasise the importance of this concept, further research has to be conducted with the aim to deliver sound and profound justifications why performance metrics are an important issue to consider. In a worst-case scenario, the end user would have to pay the price for a lacking consideration of Performance Testing in form of bad-performing or even non-working applications.

7. REFERENCES

[1] Applied performance management survey. Technical report, Compuware, 2006.

[2] Eurostat. statistical office of european union, 2013.

[3] S. Barber. Tester pi: Performance investigator, 2006.

[4] M. Brandwin. Mobile application performance engineering: A lifecycle approach to achieving confidence in application performance. 2011.

[5] M. Harman. The current state and future of search based software engineering. In *2007 Future of Software Engineering*, pages 342–357. IEEE Computer Society, 2007.

[6] M. Harman, P. McMinn, J. T. de Souza, and S. Yoo. *Search based software engineering: Techniques, taxonomy, tutorial*, pages 1–59. Empirical Software Engineering and Verification. Springer, 2012.

[7] IBM. 10 Million UK Consumers Using Mobile Commerce but 83% Have Experienced Problems. Technical report, Tealeaf, 2011.

[8] E. Lucas. Real devices and real networks: Ensuring mobile performance, 8/14 2012.

[9] Microsoft Corporation. Fundamentals of engineering for performance, 2013.

[10] S. Perez. Mobile app market: $25 billion by 2015, 1/18 2011.

[11] C. Ryan and P. Rossi. Software, Performance and Resource Utilisation Metrics for Context-Aware Mobile Applications. *11th IEEE International Software Metrics Symposium (METRICS'05)*, (Metrics):12–12, 2005.

[12] D. Schmidt. Model-driven engineering. *Computer-IEEE Computer Society*, 39(2):25–31, 2006.

[13] C. U. Smith and L. G. Williams. *Performance solutions: a practical guide to creating responsive, scalable software*. Addison Wesley Publishing Company Incorporated, 2001. 2001022849.

[14] C. U. Smith and L. G. Williams. Best practices for software performance engineering. In *CMG-CONFERENCE-*, volume 1, pages 83–92. Computer Measurement Group; 1997, 2003.

[15] C. Stary. Performance parameters and context of use. In *Performance Engineering, State of the Art and Current Trends*, pages 119–130, London, UK, UK, 2001. Springer-Verlag.

[16] C. Thompson, J. White, B. Dougherty, and D. C. Schmidt. Optimizing Mobile Application Performance with Model-Driven Engineering. pages 36–46, 2009.

[17] G. van der Heiden and P. Redshaw. Banking industry lessons learned in outsourcing testing services. Technical report, Gartner, 2012.

[18] E. Weyuker and F. Vokolos. Experience with performance testing of software systems: issues, an approach, and case study. *IEEE Transactions on Software Engineering*, 26(12):1147–1156, 2000.

[19] M. Woodside, G. Franks, and D. C. Petriu. The future of software performance engineering. In *2007 Future of Software Engineering*, FOSE '07, pages 171–187, Washington, DC, USA, 2007. IEEE Computer Society.