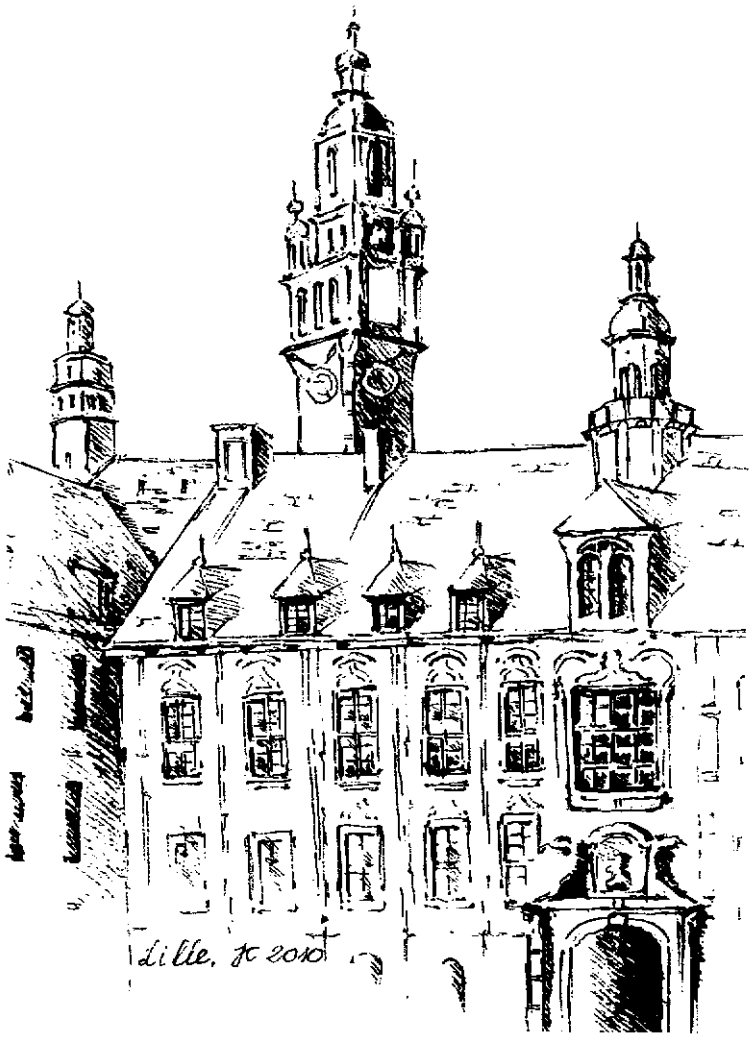

Proceedings of the Work in Progress Session

SEAA 2010

the 36th EUROMICRO Conference on Software Engineering and Advanced Applications

and

DSD 2010

the 13th EUROMICRO Conference on Digital System Design

Lille (France), September 1-3, 2010

Erwin Grosspietsch, Konrad Klöckner (eds.)

SEA-Publications: SEA-SR-27

Institute for Systems Engineering and Automation
Johannes Kepler University Linz, Austria
sec@sea.uni-linz.ac.at

SEA

JOHANNES KEPLER UNIVERSITY LINZ
Research and teaching network

ISBN 978-3-902457-27-1

Impressum

Schriftenreihe: SEA-Publications
of the Institute for Systems Engineering and Automation

J. Kepler University Linz

Proceedings of the
Work in Progress Session
held in connection with
SEAA 2010, the 36th EUROMICRO Conference on
Software Engineering and Advanced Applications
and
DSD 2010, the 13th EUROMICRO Conference
on Digital System Design

Lille (France), September 2010

Erwin Grosspietsch, Konrad Klöckner (eds.)

printed:

COREP Vauban
55, boulevard Vauban

59800 LILLE

ISBN 978-3-902457-27-1
Institute for Systems Engineering and Automation
www.sea.uni-linz.ac.at

Preface

The EUROMICRO Conference on Software Engineering and Advanced Applications and the EUROMICRO Conference on Digital System Design have organized a common Special Session to present work in progress aimed to authors that have not yet attained final and complete results in their research. The scope of the topics ranges over the full spectrum of topics of EUROMICRO conferences, e.g.

Digital Systems Design:

- Processor and memory architectures
- Special architectures
- Specification and modelling
- System validation
- System synthesis
- System-on-chip

Multimedia and Telecommunications:

- Multimedia systems
- Telecommunications
- Tools and applications

Software Process and Product Improvement:

- Software process assessment and improvement
- Organisational and business views to process improvement
- Quantitative models for development processes and products
- Distributed software development and virtual organisations
- Process and product improvement for e-business application engineering
- Verification and validation of software products
- Approaches improving dependability of software systems
- Industry best practice experiences and case studies in above areas

Component-based Software Engineering:

- Component Models
- Component Specification and Certification
- COTS (Commercial off the shelf)
- Components and Reuse
- Component Development Processes
- Component-based Architecture
- Deployment and Adaptation
- Design, Implementation, Testing
- Component Configuration Management

Our Call for Papers for the Work in Progress Session had a very good response with regard to the quantity and quality of papers, so that we were able to select 24 contributions, with authors from 9 countries. Each of these contributions consists of a short presentation in the session and an extended abstract gathered in this proceedings volume.

We thank the Johannes Kepler University of Linz, Austria for providing the publication of this volume, and all organizers of the Lille conference event for supporting the holding of the session.

Sankt Augustin, July 2010
Erwin Grosspietsch, Konrad Klöckner

SEAA 2010/DSD 2010 Work in Progress Session

Programme

Session I

Ladislav Behal, Jiri Giesl, Karel Vlcek
Chaos-Based Interleaver Design

Koki Shirakawa, Takashi Uemura, Yukihiro Iguchi
A Realization Method of Forward Converters from Multiple-Precision Binary Numbers to Residue Numbers with Arbitrary Modulus

Kenji Takahashi, Yuji Matsuda, Yukihiro Iguchi
A Realization Method of Fast Programmable Logic Controllers

Musil Tomáš, Leso Martin, Jáneš Vlastimil, Jánešová Mária
Safety Core Approach for the System with High Demands for a Safety and Reliability Design in a Partially Dynamically Reconfigurable FPGA

Martin Kohlík, Hana Kubátová
Model of Modular Secured Designs for Calculations of Availability

Jaroslav Borecky, Hana Kubátová
Dependable Interconnection of Dependable Blocks

D.Calvo, E.Villar, A.Acquaviva, E.Macii
An Approach for High-Level Thermal Modeling using Native Simulation

Vít Fábera , Vlastimil Jáneš, Mária Jánešová
Test of Genetic Algorithm with Fitness Measuring Distance between FSMs

Senol Arikan, Markus Hillenbrand, Paul Müller
A Runtime Testing Framework for Web Services

Michael Koch, Markus Hillenbrand, Paul Müller
A Monitoring Framework for the Venice Service Grid

Joachim Götze, Simon Schwantzer, Tino Fleuren, Paul Müller
Distribution of Licensed Content in Grid Environments

Tino Fleuren, Joachim Götze, Voichita Droanca, Paul Müller
Improving Data Management of Orchestrated Web Services

Session II

Chaos-Based Interleaver Design

Ladislav Behal, Jiri Giesl, Karel Vlcek
Department of Applied Informatics
Tomas Bata University in Zlin
Nad Stranemi 4511, 760 05 Zlin, Czech Republic
behal@fai.utb.cz, jgiesl@fai.utb.cz, vlcek@fai.utb.cz

1. Introduction

The interleaver plays a fundamental role in the performance of turbo coding. It was stated [1] that random-like interleavers commonly provide better performance than structured interleavers.

The chaotic systems demonstrate chaotic behavior while remaining deterministic; because of these features, they can be utilized as pseudo-random generators. Mainly because of their noise-like behavior and possibilities for security improvement (larger key space, higher diffusion characteristics), application of this type of generators attracted a vast amount of researchers who successfully applied them in source encryptions [2, 5, 6] or in channel communications [3].

Embedding the chaotic behavior into a convolutional coder was elaborated in [7], where chaotic generators were used in the form of switchers. This chaotic coding technique uses a phenomenon called "chaos synchronization" [8] which ensures that both maps in transceiver and receiver are synchronized correctly. This solution brought slight decrease of coding performance, but contributed to an increase in security.

The decrease of coding performance in [7] could be eliminated by allocation of chaotic system behavior into the interleaving process within the turbo-coding algorithm. Therefore, this paper presents a performance analysis of the chaotic interleaving.

2. Interleaver design

2.1. Utilization of chaotic maps

Let a discrete dynamical system be described by the following simple formula (1):

$$x_{k+1} = f(x_k), \qquad f: I \to I, \qquad x_0 \in I \tag{1}$$

where f is a continuous map on the interval $I = [0,1]$.

This system can be referred to as a chaotic one if the conditions for topological transitivity as well as for sensitivity to initial conditions are satisfied [4].

2.2. Chaotic interleaver design

The interleaver can be implemented as an array of integer values. These integers represent permutation rules (positions) for data interleaving. The proposed chaotic interleaver generates these integer values by using chaotic maps.

Let $\pi(i)$ be permutation mapping at time i, f^n the output of the chaotic map in n-th iteration and T be the size of the interleaver. The generation of π could be described by the following formula:

$$\pi(i) = f^n \cdot T \tag{2}$$

where $i = n = 1, 2, \ldots T$. All values in π are rounded to integer values.

The experimental results for three different chaotic-interleavers and five classical interleavers are compared in the next chapter. Three chaotic interleavers use logistic, cusp or cubic map [4].

We have chosen these particular maps for their low computation complexity costs. Further, these maps can produce highly complex behavior when interconnected in parallel connection. The chosen maps can be described by following formulas (3-5):

1. Logistic map
$$X_{n+1} = A \cdot X_n \cdot (1 - X_n) \tag{3}$$
2. Cubic map
$$X_{n+1} = A \cdot X_n \cdot (1 - X_n^2) \tag{4}$$
3. Cusp map
$$X_{n+1} = 1 - A \cdot \sqrt{|X_n|} \tag{5}$$

The parameter A is the so-called "control parameter" which influences the orbit state [4]; and therefore, A must be set accordingly so as to satisfy the conditions for topological transitivity and sensitivity to initial conditions described in [4].

inspection results and defect distributions in the code parts under test. For example, an assumption can be: "Code classes in which a high number of inspection defects are found indicate more defects to be found with testing." Regarding defect types, an assumption can be that defect types which are found with the inspection are also expected to be found with testing. Here, assumptions to be applied in a concrete context should be based on hypotheses empirically validated or derived in a context-specific manner. Next, the general assumptions need to be refined in a systematic manner into selection rules in order to be applicable. A concrete selection rule might be: "Focus your test activity on those two defect types which are found most often based on the inspection results." Another selection rule that combines a metric gathered from the code with the inspection result can be: "Focus your test activity on all code classes where the defect content is higher than 30 defects per 1000 lines of code based on the inspection results and where the class size is larger than 700 lines of code." To obtain reliable inspection results, quality monitoring has to be performed, which can, for example, be based on historical data from the context. Finally, concrete test cases can be derived for the predicted code classes or to cover the expected defect types.

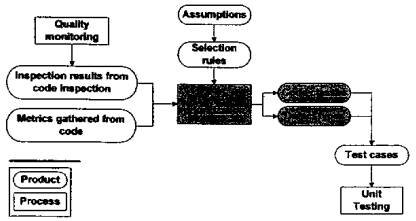

Figure 1: Concepts of the integrated inspection and testing approach with respect to code

3. Conclusion

We motivated the need for test effort reductions and gave a brief overview of different existing approaches that prioritize test activities in order to reduce test effort. However, defect data from current or early performed QA activities are often not used for prioritization. Thus, we propose an integrated inspection and testing approach that explicitly uses defect data from an inspection to prioritize test activities. We do not claim that inspection data is the only appropriate predictor of defects or defect types, but it can give valuable support for allocating test effort and thus, to reduce test effort. Moreover, not only the test effort, but the overall QA effort may be reduced by the approach.

Next, we want to evaluate the integrated approach and use the results to enhance and refine the current approach. Besides checking the applicability of our approach, our main research goal is to evaluate whether the integrated inspection and testing approach leads to an effort reduction for testing at a comparable quality level compared to a non-integrated approach. Moreover, we plan to evaluate different assumptions and selection rules in a given environment to obtain first insights into which ones might lead to suitable results for prediction of defect-prone code classes and expected defect types.

ACKNOWLEDGMENT

We thank Sonnhild Namingha for proofreading.

REFERENCES

[1] N. Juristo, A.M. Moreno, S. Vegas, "Reviewing 25 years of testing technique experiments," Emp. Software Engineering, pp. 7-44, 2004

[2] A. Aurum, H. Petersson, C. Wohlin, "State-of-the-art: software inspections after 25 years," Software Testing, Verification and Reliability Journal, pp. 133-154, 2002

[3] R. Pressman. Software engineering: a practitioner's approach, McGraw-Hill, London, 5th edition, 2000

[4] M. Harrold, "Testing: A Roadmap", Int. Conference on Software Engineering, Limerick, Ireland, pp. 61-72, 2000

[5] V.R. Basili, B.T. Perricone, "Software errors and complexity: an empirical investigation," Com. of the ACM, pp. 42-52, 1984

[6] T.J. McCabe, "A complexity measure," IEEE Transactions on Software Engineering, pp. 308-320, 1976

[7] T.J. Ostrand, E.J. Weyuker, "The distribution of faults in a large industrial software system," International Symposium on Software Testing and Analysis, pp. 55-64, 2002

[8] N. Nagappan, T. Ball, A. Zeller, "Mining metrics to predict component failures," International Conference on Software Engineering, pp. 452-461, 2006

[9] N.E. Fenton, N. Ohlson, "Quantitative analysis of faults and failures in a complex software system," IEEE Transactions on Software Engineering, pp. 797-814, 2000

[10] A. Schröter, T. Zimmermann, R. Premraj, A. Zeller, „If your bug database could talk," 5th Int. Symposium on Empirical Software Engineering, pp. 18-20, 2006

[11] T. Illes-Seifert, B. Paech, "Exploring the relationship of a file's history and its fault-proneness: An empirical method and its application to open source programs," Information and Software Technology Journal, pp. 539-558, 2009

[12] H. Pham, System Software Reliability, Springer, 2006

[13] N. Nagappan, T. Ball, "Static analysis tools as early indicators of pre-release defect density," 27th international conference on Software engineering, pp. 580-586, 2005

[14] Findbugs, available: http://findbugs.sourceforge.net/

[15] I. Holden, D. Dalton, "Improving testing efficiency using cumulative test analysis," Testing: Academic & Industrial Conference on Practice and Research Techniques, pp. 152-158, 2006

[16] D. Zhang, C. Nie, "A markov decision approach to optimize testing profile in software testing," 9th International Conference for Young Computer Scientists, pp. 1205-1210, 2008

[17] Orthogonal Defect Classification, IBM, available: http://www.research.ibm.com/softeng/ODC/ODC.HTM

Defect Detection Using Event-Based Process Analysis in (Software+) Engineering Projects

Wikan Danar Sunindyo, Stefan Biffl, Christian Frühwirth, Richard Mordinyi, Thomas Moser, Alexander Schatten, Sebastian Schrittwieser, Edgar Weippl, Dietmar Winkler
Christian Doppler Laboratory for
Software Engineering Integration for Flexible Automation Systems
Vienna University of Technology, Austria
{wikan, biffl, tmoser, schatten, weippl, winkler} @ifs.tuwien.ac.at

1. Introduction

Modern software-based systems, like industrial automation systems typically involve the cooperation of several engineering fields, e.g., mechanical, electrical, and software engineering [1]. We call this kind of cooperation "(software+) engineering projects", as software engineering increasingly provides added value to the resulting software-intensive systems and also depends on the seamless collaboration with all other systems engineering disciplines. In (software+) engineering projects a wide range of heterogeneous tools and data models are used by the engineers, which make early defect detection challenging due to semantic and technical gaps between the selected tools and data models. An integration platform, like the engineering service bus [1], can provide the foundations to overcome these gaps but needs to be extended with a process-oriented view for effective defect detection.

To assure and improve the quality of the engineered system, project and quality managers need a comprehensive view on verification and validation of the system. Currently, individual and selective quality assurance (QA) methods are applied to the system with only limited scope. This approach is time-consuming and requires acceptance tests, which identify defects late in the engineering process. If the system is changed, the QA applied to detect defects as results of these changes is often insufficient and not systematic.

Major challenges in (software+) engineering quality management include engineering model version and change management, early defect detection across engineering discipline and tool boundaries, and engineering process analysis to identify the sources of defects.

In this paper, we address one of these challenges, identifying sources of defects in (software+) engineering projects, by analyzing event data coming from (software+) engineering projects. For an initial evaluation, we study defects in the use case "continuous integration and test" (CI&T), a standard software engineering (SE) process that is implemented rigidly in modern SE environments.

2. Related Work

Process analysis approach has been applied to engineering systems, like workflow management systems, Enterprise Resource Planning (ERP) and Customer Relationship Management (CRM) [2]. Schonenberg et al. continue the research on performance analysis based on process mining by proposing a recommendation service to the Process Aware Information Systems (PAISs) [3]. This approach is based on similar past process executions by considering the specific optimization goals. Risk analysis to experimental results of the system can also be applied to detecting defects in (software +) engineering processes.

3. Research Issues

Technical and semantic integration provide the foundation for integrated QA of engineering processes in (software) engineering [1] to detect defects earlier in the engineering process. The next task is to investigate a process analysis approach to detect defects across different engineering disciplines and across tool boundaries that builds on this foundation.

We will discuss the requirements for (software+) engineering process analysis and how it can facilitate the detection of defects and their sources. The tasks include defining the types of defects to be collected, the collection of suitable process data (e.g., engineering tool events), the aggregation and transformation of the collected process data and finally providing the result of the defect detection based on the process analysis approach.

4. Solution Approach

Process Analysis Requirements. In order to analyze engineering processes of (software+) engineering systems, we need to define first the use case to which we apply the experiment and perform the analysis. This is required since certain detail tasks to analyze could differ from one use case to other use cases. Use case definition may simplify the setting of the goals of the analysis, e.g., what kind of defects should be addressed (e.g., detected, localized, or recovered). Process data from the system is also required for analysis. These data can be collected from event logs or communication data between different engineering fields of the system (e.g., data from chat server, mail server, etc.). Some integration and transformation needs to be applied to the data in order to get clean and well-formed data for further analysis. Finally, methods and tools to analyze data need to be chosen and applied.

CI&T Use Case. Figure 1 illustrates an overview of the proposed defect prediction approach using event-based process analysis. Event data originating from heterogeneous engineering tools is collected and integrated in the event log (3) by using an Engineering Knowledge Base [4] (2) and the Engineering Service Bus [1] (1). By using a process analysis tool [5] (4), we can provide information for detecting potential sources of defects (5) in (software+) engineering project environments.

The CI&T use case is a standard life cycle process for SE consisting of several steps: (a) On change of a code unit, build the source code by using a build server, (b) Test the built source code by using a test tool, (c) Package and deploy the compiled source code by using a source code management system. The build result will be published on a project website and if there are errors, a notification will be sent to a list of recipients. This process is implemented by integration build served like Hudson or Continuum in a rigid way.

5. Conclusion

Current status of the work is the building of research prototypes of relevant engineering process support systems and analyzing the results of systematic test runs. Early results with the CI&T process indicate that process analysis is helpful fort testing distributed engineering systems that use a common infrastructure for collecting relevant process events. In addition to tracing the expected process flows in the analysis, we also found symptoms of unexpected behavior that helped find defects in the engineering system.

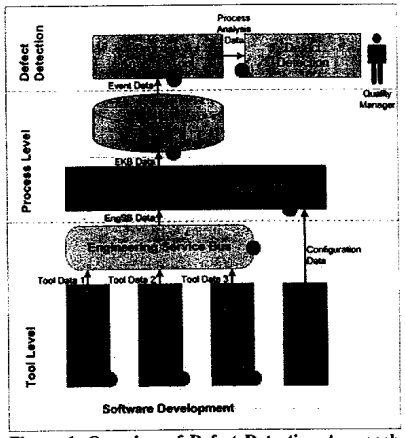

Figure 1. Overview of Defect Detection Approach Using Event-Based Processes Analysis [5]

Acknowledgments

This work has been supported by the Christian Doppler Forschungsgesellschaft and the BMWFJ, Austria.

References

[1] S. Biffl, and A. Schatten, "A Platform for Service-Oriented Integration of Software Engineering Environments," in Eight Conference on New Trends in Software Methodologies, Tools and Techniques (SoMeT 09), 2009, pp. 75-92.

[2] W. M. P. van der Aalst, A. J. M. M. Weijters, and L. Maruster., "Workflow Mining: Discovering Process Models from Event Logs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 9, pp. 1128-1142, 2004.

[3] H. Schonenberg, B. Weber, B. Dongen, and W. Aalst, "Supporting Flexible Processes through Recommendations Based on History," in Proceedings of the 6th International Conference on Business Process Management, Milan, Italy, 2008.

[4] T. Moser, S. Biffl, W. D. Sunindyo, and D. Winkler, "Integrating Production Automation Expert Knowledge Across Engineering Stakeholder Domains," in 4th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS 2010), Krakow, Poland, 2010, pp. 352-359.

[5] W. D. Sunindyo, T. Moser, D. Winkler, and S. Biffl, *Foundations for Event-Based Process Analysis in Heterogeneous Software Engineering Environments (Technical Report)*, Christian Doppler Laboratory for Software Engineering Integration for Flexible Automation Systems, Vienna, Austria, 2010, http://www.ifs.tuwien.ac.at/files/u290/Foundations_for_Event-Base_Process-techrep.pdf.

Model Checking for Generation of Test Suites in Software Unit Testing

Vasilios Almaliotis, Panagiotis Katsaros, Konstantinos Mokos
Department of Informatics
Aristotle University of Thessaloniki
54124 Thessaloniki, Greece
{valmalio, katsaros}@csd.auth.gr, mokosko@otenet.gr

I. Introduction

Reports on software economics mention that testing plays a significant role in software development, since more than 50 percent of the total cost of a software project is often expended in testing. Although, software testing can show the presence of bugs, it is inadequate for showing their absence and requires highly skilled engineers. Traditionally, software testing and model checking are dealt as separate verification and validation activities. However, recent works invest on the potential of model checking towards reducing the cost of software testing [1, 2]. The common denominator of these techniques is the development of an appropriate set of linear temporal logic (LTL) formulae that when applied to the program model produce a set of test cases for a given coverage criterion.

Specifically, in [1] the authors try to reduce the cost for test case generation by employing heuristic algorithms in order to minimize the set of LTL formulae in the test suite. The syntax of the used LTL formulae is relatively straightforward for a human being, but their production is not easily automated [2].

In this article we focus on the automated extraction of a model program from the source, in an attempt to reduce the cost for the generation of test cases for unit testing. The created model program is constructed exclusively for test case generation, which means that additional information is embedded and the program undergoes suitable transformations, in order to aid the specific model checking process that yields the expected test cases. By abstracting this information at the level of the model program we eliminate the need to specify it in the LTL formulae. In this way, the needed LTL formulae are kept as simple as possible and therefore we can support end-to-end automation from the source program to the generation of the expected test cases.

II. Model Checking For Generation Of Test Suites In Software Unit Testing

The proposed method for automatic generation of test cases in software unit testing relies on a Kripke structure representation of the model program and is performed in two phases as shown in Figure 1.

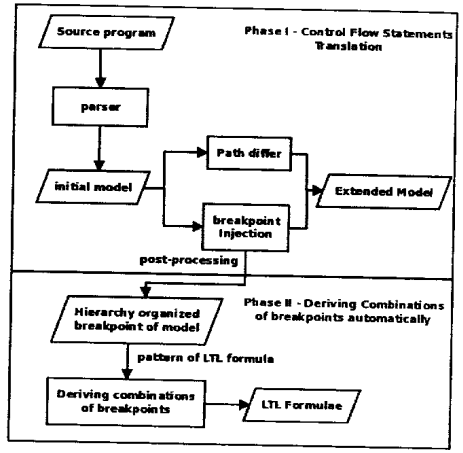

Figure 1. Proposed Method Process

During the first phase, the software unit is parsed and abstracted into an initial model program. The initial model program is enhanced with breakpoints and possible execution paths are thus differentiated during the automated translation of the flow statements to the control flow constructs of the extended model program. The model program is expressed in PROMELA, which is the input language of the SPIN model checker [3]. In the second phase, hierarchically organized breakpoints of the extended model program are selected based on the chosen coverage criterion (post-processing) that was determined during the path differentiation process of the previous phase. By automatically deriving combinations of breakpoints an appropriate LTL formula is created. The counterexamples obtained from model checking the generated LTL formulae, form the test suite for the coverage criterion at hand. The prototype tool support that is currently available implements the multiple condition coverage criterion [4], which is considered as the most comprehensive control flow coverage alternative. It is relatively straightforward to implement a less costly coverage criterion such as edge/condition coverage but in the current prototype this requires

EAL7: Formally Verified Design and Tested – Applies to the development of security targets of evaluation for application in extremely high-risk situations, as well as when a high value of the assets justifies the higher costs.

4. Security Factor in FPA

Part of the software project analysis will be a security evaluation. The level of security, which the final product has to fulfil, is determined. The security evaluation of software will be determined in compliance with software security criteria described in chapter 3 – *Common Criteria*. Depending on the level of security the influence of cost estimation of software products will be defined. The new factor – *Security of Product* is added among factors of technical and operational complexity, such as the 15th factor. As well as the other 14 factors, the security factor is evaluated by a six-point scale 0 – 5 according to its effect on the application.

15. General System Characteristic – *Security of Product:*
    - *Definition:* Security of Product describes the degree of security of the application in compliance with the security evaluation criteria – Common Criteria.
    - *Score:*

Score As	Description To Determine Degree of Influence
0	EAL1
1	EAL2, EAL3
2	EAL4
3	EAL5, (EAL4)
4	EAL6
5	EAL7 (EAL6)

Table 1: Degree of influence of new factor

Consequently the equation which is used for the determination of an adjusted function points count is modified:

$$FP = UFP \cdot VAF$$

$$UFP = \sum EIw + \sum EOw + \sum EQw + \sum ILFw + \sum EIFw$$

$$VAF = TDI \cdot 0.01 + 0.65$$

VAF is the value adjusted factor; *UFP* is the count of unadjusted function points; *w* is weight of each function type; and *TDI* (total degree of influence) is the factor of technical and operational complexity – calibration parameter of the workload, which demonstrates the effect of 15 factors, out of which each is evaluated in a six-point scale 0 – 5 according to its effect on the application (0– without effect, 1 – accidental effect, 2 – simple effect, 3 – average effect, 4 – complex effect, 5 – substantial effect). *TDI* is determined as an addition of the given estimation of all mentioned factors listed below, where F_i is the weight of attribute 'i'.

$$TDI = \sum_{i=1}^{15} F_i$$

The considered factors are: Data communication; Distributed data processing; Performance; Heavily used configuration; Transaction rate; On-line data entry; End-user efficiency; On-line update; Complex processing; Reusability; Installation ease; Operational ease; Multiple Sites; Facilitate change; Security. On the basis of the determined value of adjusted function points, the workload intensity, development time and total costs have to be calculated.

5. Conclusion

Costs estimation which is needed for the realization of a software project is considered to be common activity. Software products must be sufficiently protected in order to resist various types of attacks. Consequently and proportionate to the level of protection of a software product, the requirements for labour input increases. Therefore, it is necessary to analyse the level of protection of a system already in the cost estimation phase. On this basis, the appropriate labour input for security can be calculated as part of the total cost of software product development. The software security field is a relatively new one and most of the cost estimation models do not consider the costs that invoke security while developing software. The idea of this article was modified where discussion of the Function Point Analysis included costs that invoke security to workload intensity, development time and total costs of software project development. At present, the analysis of the influence of the various levels of security to total costs is realized and evaluated on real data.

Acknowledgement: This research was supported by the BUT FIT grant FIT-S-10-1 and the Research Plan No. MSM, 0021630528 Security-Oriented Research in Information Technology.

References

[1] IFPUG - International Function Points User Group. 2010. Function Point Counting Practices Manual. Release 4.3.1. Princeton Junction: IFPUG, 2010. ISBN 978-0-9753783-4-2.

[2] Doucek, J.; Novák, L.; Svatá, V.: Řízení bezpečnosti informací. Příbram, Professional Publishing 2008, ISBN 978-80-86946-88-7.

[3] Parthasarathy, M. A.: Practical Software Estimation: Function Point Methods for Insourced and Outsourced Projects, Addison-Wesley Professional 2007, ISBN 0-321-43910-4.

[4] Jaquith, A.: Security Metrics: Replacing Fear, Uncertainty, and Doubt, Addison-Wesley Professional 2007, ISBN 0-321-34998-9.

Research Challenges in the Security Design and Evaluation of an Engineering Service Bus Platform

Christian Frühwirth[1] Stefan Biffl[2] Alexander Schatten[2] Sebastian Schrittwieser[2] Edgar Weippl[2]

[1]*Aalto University*

[2]*Vienna University of Technology - Christian Doppler Laboratory for Software Engineering Integration for Flexible Automation Systems*

Christian.Fruehwirth @ tkk.fi

{Stefan.Biffl, Alexander.Schatten, Edgar.Weippl} @ tuwien.ac.at, Sschrittwieser @ sba-research.org

1. Introduction

The Open (Software) Engineering Service Bus (EngSB)[6] is based on the Apache ServiceMix ESB and connects software tools across engineering domains and company borders. EngSB allows the integration and automation of engineering processes by connecting multiple tools in seamless information workflows through the use of tool connectors, domain bridges, intelligent service message transformation and routing [3][4][5]. Figure 1 gives an overview of the EngSB application architecture.

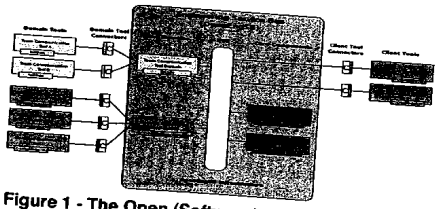

Figure 1 - The Open (Software) Engineering Service Bus Architecture

While the benefits of an engineering service bus application are attractive for companies and researchers, they come with a number of open challenges on the security side. Even though application- and service security has received tremendous attention from the security research community in the past, the engineering service bus approach represents a novel concept that is not yet sufficiently understood on the security level. Thus, before companies can fully adopt and realize the potential of engineering service-bus concepts, more work has to identify and address these open security research issues. This paper aims to guide such future work by proposing a set of research issues in the EngSB context. The presented issues will provide valuable support in the targeted future research work.

2. Security research challenges

We use Schneier's Security Decision Model[5], shown in Figure 2 to determine the areas of open research issues.

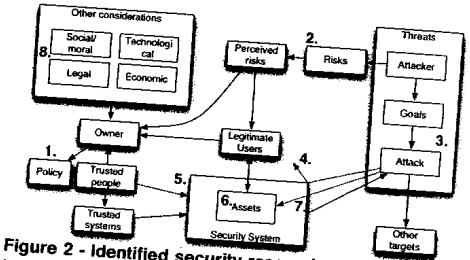

Figure 2 - Identified security research issues, placed in Schneier's Security Decision Model [5]. Numbers indicate the interaction area of the identified issue.

Schneier's security decision model is useful because it gives a comprehensive overview of the relations and interactions between information assets, the security system that is set to protect them and the outside world with its various stakeholders. We identified 8 main security research issues by applying the model to the EngSB context: **Issue 1.: Developing a security policy for a Service Bus that integrates economic considerations and competing stakeholder values:** An EngSB operates in a complex, distributed environment that spans across engineering domains and companies with conflicting values, thus the inherent risks are less clear than in traditional isolated systems and economic aspects have a strong influence on policy decisions[1][5]. Research work needs to address this issue and find new ways to align competing stakeholder values with quality requirements and economic realities [7]. **Issue 2.: Determining risk exposure to enable a value-based management of security:** Understanding the relationship between attackers, their motives as well as the differences between perceived and actual risk exposure is essential for value based security management. Little research has yet been done that combines these relationships with the inter-stakeholder dependencies on a service bus under the concept of value based security. **Issue 3.: Improve quality of empirical security data to move from defensive to attacker-focused countermeasures:** Traditional security focuses on individual defense (build high walls around everything). Since this is economically unvi-

able in a large, heterogeneous system, a clearer understanding of the attacker and its motives is needed to better target countermeasures (build high walls only around the gold chest) and increase efficiency (build walls only slightly higher than a man can jump). Today's research in this area faces a severe lack of empirical data on attackers and their motives since companies that fall victim to an attack do typically not disclose such information. Future work should address the empirical side of this problem by improving the collection, reliability and communication of both, attack and countermeasure related data. **Issue 4.: Evaluating the effectiveness of a security system with security metrics:** The effectiveness of a security system is difficult to measure without a baseline for data comparison. Research in this area would need to determine an empirically valid baseline and set the guidelines for its evaluation. There are numerous security metrics available to evaluate security systems; however, they need to be improved and tailored for the application in a service bus setting. Little research has yet been done on the effectiveness of such metrics, if/how they help developers to improve software quality and how to choose the right metric for a given evaluation task. **Issue 5.: Seamless integrating a security system in the information workflow:** Too many developers still consider security an "add-on" feature. This detachment of security from the overall application can lead to interruptions or breaks in the information workflow between the users and the system. System architects thus face the question of how to make better use of "security by design" principles in order to achieve a seamless integration of security in the overall information workflow. There has not yet been extensive research on how much "seamless" security can contribute to the performance of a business process, i.e. how much it would be worth to pay for. **Issue 6.: Asset management and valuation:** In order to manage the security of information assets in a system effectively, the nature and value of the assets needs to be know. On a distributed and dynamic platform like the EngSB information assets can change rapidly, thus static asset management is not enough. Even though there are existing solutions to manage inventories of information assets, the problem of putting a value or price tag on a changing asset remains. **Issue 7.: Security Incident management on a service bus:** A security incident in an integrated system can jeopardize the operations of several connected companies at the same time, with consequences hard to predict. In order to manage and remediate the risk from security incidents, the EngSB platform needs to find ways of developing survivability capabilities. On the policy level, existing incident management frameworks (like parts of ITIL [8]) need to be adapted to suit the EngSB environment. Adapting existing frameworks, however, raises the question of how these modifications would align with stakeholder requirements, and the overall EngSB (software) architecture. There is yet little research on the integration of the software side of security incidents with the business side (e.g. how can an organization continue to function when the bus that connects its tools is out of order). **Issue 8.: Continuous compliance:** The target users of an engineering service-bus are mostly larger companies which typically fall under some form of compliance regulations like the Sarbanes Oxley Act (SOX). Compliance regulations dictate major parts of organizations' IT service management (ITSM) and security policies, thus also the way companies will (or can) use EngSB. In order for service busses like the EngSB to become successful on a wider scale, future work should investigate whether security design in the EngSB architecture can facilitate companies' compliance efforts.

3. Conclusion

We used Schneier's Security Decision Model, to determine areas of open research issues in the security design and evaluation of an engineering service bus. Many of the described issues pointed towards increasingly value-oriented security approaches that will enable application developers and companies to make better decisions on how and where to target their security efforts. We believe that addressing these open research issues will encourage future work in this area and guide improvement efforts in directions that improve the quality and application of engineering service bus concepts.

References

[1] R. Anderson and T. Moore, "The economics of information security," *Science*, vol. 314, 2006, pp. 610-613.

[2] S. Biffl, S.B. Schatten, and A. Zoitl, "Integration of Heterogeneous Engineering Environments for the Automation Systems Lifecycle," *Proc. IEEE Industrial Informatics Conf*, 2009, pp. 576-581.

[3] S. Biffl, W.D. Sunindyo, and T. Moser, "Bridging Semantic Gaps Between Stakeholders in the Production Automation Domain with Ontology Areas," *Proceedings The 21st International Conference on Software Engineering Knowledge Engineering (SEKE 2009)*, 2009, pp. 233–239.

[4] T. Moser, S. Biffl, W.D. Sunindyo, and D. Winkler, "Integrating Production Automation Expert Knowledge Across Engineering Stakeholder Domains," *2010 International Conference on Complex, Intelligent and Software Intensive Systems*, 2010, pp. 352–359.

[5] B. Schneier, "Nonsecurity considerations in security decisions," *IEEE Security & Privacy*, 2007, p. 88.

[6] The Open Engineering Service Bus, available online at http://openengsb.org, 2010.

[7] N. Oza, S. Biffl, C. Frühwirth, Y. Selioukova, and R. Sarapisto, "Reducing the Risk of Misalignment between Software Process Improvement Initiatives and Stakeholder Values," Industrial Proceedings of EuroSPI, 2008, pp.6–9.

[8] ITIL, "The Open Guide. ITIL Incident Management" Available online at: www.itlibrary.org , 2010

Implementing Brute-Force Attack on PRESENT Cipher

Pavel Benáček
Faculty of Electrical Engineering
Czech Technical University in Prague
Karlovo nám, 13, 121 35 Praha 2, Czech Republic
Email: benacpav@fel.cvut.cz

Martin Novotný
Faculty of Information Technology
Czech Technical University in Prague
Kolejní 550/2, 160 00 Praha 6, Czech Republic
Email: novotnym@fit.cvut.cz

I. Introduction

Lightweight cryptography finds many applications in nowadays products. SmartCards, RFID tags or remote keyless entry systems require encrypted communication. Unfortunately, many of such products suffer from cryptanalytically weak ciphers and/or implementations. Variety of attacks against ciphers like Mifare (Crypto1), KeeLoq or Hitag-2 have been published in open literature.

The cipher PRESENT [1] has been designed as a replacement of above mentioned weak ciphers. It offers high cryptographic strength, while it enables space-saving hardware implementation. PRESENT is a block cipher with a block size of 64 bits and a key size of either 80 or 128 bits. It is based on the Feistel network with 32 rounds. The implementation of the cipher can be scaled from small area architecture useful for lightweight cryptography to large area pipelined architecture offering high throughput.

In this work we present our analysis of resistance of the PRESENT cipher against the brute-force attack. We have chosen its 80 bit variant. As a target platform we have chosen server COPACOBANA.

A. Implementation Platform

The COPACOBANA (Cost-Optimized Parallel Code Breaker) machine [2], [3] is a high-performance, low-cost cluster consisting of 120 Xilinx Spartan3-XC3S1000 FPGAs. Currently, COPACOBANA appears to be the only such reconfigurable parallel FPGA machine optimized for code breaking tasks reported in the open literature. Depending on the actual algorithm, the parallel hardware architecture can outperform conventional computers by several orders of magnitude. COPACOBANA has been designed under the assumptions that (i) computationally costly operations are parallelizable, (ii) parallel instances have only a very limited need to communicate with each other, (iii) the demand for data transfers between host and nodes is low due to the fact that computations usually dominate communication requirements and (iv) typical crypto algorithms and their corresponding hardware nodes demand very little local memory which can be provided by the on-chip RAM modules of an FPGA. Considering these characteristics COPACOBANA appeared to be perfectly tailored for simple brute-force attack on PRESENT like the one described in the next section.

II. Attack Architecture

The attack works with a pair of a known plaintext and a corresponding ciphertext. It is based on trial encryptions of the plaintext with all potential keys. Whenever the obtained ciphertext matches the known one, the right key is found. Presented attack is a modification of a brute-force attack on Data Encryption Standard (DES) described in [2], [3].

The block scheme of an attack engine placed in one FPGA is shown in Figure 1. It contains two encryption units (PRESENT 1 and PRESENT 2) running in parallel. The verified keys are applied to the inputs of the units. Known plaintext is encrypted under tested keys and obtained ciphertexts are compared with the known one. To gain the maximum performance, the encryption units as well as comparators are pipelined. As a consequence, each FPGA can verify two keys per one clock cycle.

The attack is controlled by a host computer connected to COPACOBANA. The host computer can either communicate with each individual FPGA or can broadcast data to all FPGAs.

During initialization the values of plaintext and ciphertext are stored in each FPGA in corresponding registers. To parallelize the attack, the 80 bit keyspace is divided into 2^{39} key subspaces defined by upper 39 bits of a key. During runtime, the host computer successively assigns every FPGA with a unique key subspace to search in. Bits defining the key subspace are stored in a corresponding register. Lower bits of keys in a subspace are generated by a 40 bit counter. The remaining 1 bit identifies one of 2 encryption units.

If the key subspace is fully searched without any success, the FPGA is assigned with another key subspace (search in one key subspace takes about 2.5 hours). If the key is found, it is sent to the host computer via communication interface.

III. Implementation Results

The brute-force engine depicted in Figure 1 occupies 73 % of chip area. The maximum achieved frequency is 120 MHz. As each FPGA can verify 2 keys per one clock cycle and the COPACOBANA is equipped with 120 FPGAs, the maximum performance is $2.88 \cdot 10^{10} \approx 2^{34.75}$ tested keys per second.

The EUROMICRO Conference on Software Engineering and Advanced Applications and the EUROMICRO Conference on Digital System Design have organized a common Special Session to present work in progress aimed to authors that have not yet attained final and complete results in their research. The scope of the topics ranges over the full spectrum of topics of EUROMICRO conferences:

Digital Systems Design
Internet technologies, quality of service and applications
Software Process and Product Improvement
Component-based Software Engineering

The Call for Papers for the Work in Progress Session had a very good response with regard to the quantity and quality of papers, so that we were able to select 24 contributions, with authors from 9 countries. The extended abstract are collected in this proceedings volume.

Johannes Kepler University Linz
Institute for Systems Engineering and
Automation
www.sea.uni-linz.ac.at

ISBN 978-3-902457-27-1