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...

ISBN 978-3-902457-27-1
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.

Sanki Augustin, July 2010
Erwin Grosspietsch, Konrad Klöckner
Session 1

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, Jiři Vlastimil, Jana 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 Borecký, 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 Jáněš, Mária Jánestová  
Test of Genetic Algorithm with Fitness Measuring Distance between FSMs

Senol Arikian, 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 Götzze, Simon Schwantzer, Tino Fleuren, Paul Müller  
Distribution of Licensed Content in Grid Environments

Tino Fleuren, Joachim Götzze, Voichita Drosuca, Paul Müller  
Improving Data Management of Orchestrated Web Services
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@fi.sai.utb.cz, jgiesl@fi.sai.utb.cz, vlcek@fi.sai.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
taught 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 switches. This chaotic coding
technique uses a phenomenon called “chaos
synchronization” [8] which ensures that both maps in
transmitter 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):

\[ x_{n+1} = f(x_n), \quad f : I \rightarrow I, \quad x_0 \in I \quad (1) \]

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 \( i \), \( j^* \) the output
of the chaotic map in \( n \)-th iteration and \( T \) be the size of
the interleaver. The generation of \( \pi(i) \) could be described by
the following formula:

\[ \pi(i) = j^* \cdot T \quad (2) \]

where \( i = 1, 2, ..., T \). All values in \( \pi \) 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 maps [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

\[ x_{n+1} = \lambda x_n (1-x_n) \quad (1) \]

2. Cubic map

\[ x_{n+1} = \lambda x_n - x_n^3 \quad (3) \]

3. Cusp map

\[ x_{n+1} = 1 - \lambda \sqrt{x_n} \quad (4) \]

The parameter \( \lambda \) is the so-called “control parameter”
which influences the orbit state [4], and therefore, \( \lambda \) must
be set accordingly so as to satisfy the conditions for
(topological) transitivity and sensitivity to initial conditions
described in [4].
Defect Detection Using Event-Based Process Analysis in (Software+) Engineering Projects


Defect detection and test generation is an important area in software engineering. Defects are often the result of human error or limitations in the development process. The main research goal is to evaluate whether the integrated inspection and testing approach leads to an improvement in the quality of the software. The approach aims to detect defects at an early stage in the development process, thus reducing the overall cost of software development. The approach involves the use of event-based process analysis, which allows for the identification of patterns and anomalies in the software development process. The results of the analysis can be used to improve the quality of the software and to reduce the cost of defects. The approach was implemented in a software engineering project at the University of Technology, Austria, and the results were presented at the International Conference on Software Engineering. The paper concludes with a discussion of the implications of the findings and future research directions.

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 other systems engineers. 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 integrated 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 a 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, 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 test case "continuous integration and test" (CI&I), a standard software engineering (SSE) process that is implemented rigidly in modern SE environments.

2. Related Work
Process analysis approaches have been applied to engineering projects, like work-flow management systems, Enterprise Resource Planning (ERP) and Customer Relationship Management (CRM) [2]. Schobenberg et al. continue the research on process analysis based on process mining by proposing a recommendation service to the Process Aware Information Systems (PAIS) [3]. This approach is based on past process execution by considering the specific optimization goals. Risk analysis to experimental results of the system can also be applied to detecting defects in software engineering projects.

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. This case definition may simplify the setting of the goals of the analysis, e.g., what kind of defects should be detected (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 a 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/CT 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) and the Engineering Service Bus (3) (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/CT use case is a standard life cycle process for SE consisting of several steps: (a) On change of a code unit, build 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 server like Hudson or Continuum as a rigid way.

5. Conclusion

Current state 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/CT process indicate that process analysis is helpful for test 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 find symptoms of unexpected behavior that helped find defects in the engineering system.

Acknowledgments

This work has been supported by the Christian Doppler Forschungsstillwerk and the BMWF, Austria.

References


Figure 1. Overview of Defect Detection Approach Using Event-Based Process Analysis

Model Checking for Generation of Test Suites in Software Unit Testing

Vasilios Almaliotis, Panagiotis Katsaros, Konstantinos Moskos
Department of Informatics
Aristotle University of Thessaloniki
54124 Thessaloniki, Greece
(almalio, katsaros)@csd.auth.gr, moskos@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 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 SUITE 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 suit 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 test suite coverage criterion such as edge/condition coverage but in the current prototype this requires
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 fulfill, 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 of Security of Product describes the degree of security of the application in compliance with the security evaluation criteria – Common Criteria.
   - Score:

<table>
<thead>
<tr>
<th>Score As</th>
<th>Description To Determine Degree of Influence</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>EAL1</td>
</tr>
<tr>
<td>1</td>
<td>EAL2, EAL3</td>
</tr>
<tr>
<td>2</td>
<td>EAL4</td>
</tr>
<tr>
<td>3</td>
<td>EAL5 (EAL4)</td>
</tr>
<tr>
<td>4</td>
<td>EAL6</td>
</tr>
<tr>
<td>5</td>
<td>EAL2 (EAL6)</td>
</tr>
</tbody>
</table>

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:

\[ PF = UFP \times VAF \]

\[ UFP = \sum Ew + \sum Ew0 + \sum EwL + \sum EwF \]

\[ VAF = TDI \times 0.01 + 0.65 \]

| VAF is the adjusted factor | UFP is the count of unadjusted function points | Ew is weight of each function type | TDI (total degree of influence) is the factor of operational and operational complexity – calibration parameter of the workload, which demonstrates the efi of each factor on the application 0 – 5 according to its effect on the application (0- without effect, 1 - incidental effect, 2 - simple effect, 3 - average effect, 4 - complex effect, 5 - substantial effect). |

The considered factors are: Data communication; Distributed data processing; Performance; Heavily used configuration; On-line data entry; End-user efficiency; On-line update; Complex process; Processability; Installation ease; Operational ease; Multiple Sites; Facilities 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 an essential 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 labor 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 labor 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 software development 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


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

While the benefits of an engineering service bus application are attractive for companies and researchers, they have a number of open challenges on the security side. Even with thorough 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 be done to identify and address these open security research issues. This paper aims to give some 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 Schneider’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 Schneider’s Security Decision Area Model[5]. Numbers indicate the interaction area of the identified research issues.

Schneider’s 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 main risk and 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 various 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][3]. 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: Breach 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 attacking strategies: Quantitative measures: Traditional security focuses on individual defenses (build high walls around everything). Since this is economically invi-
Implementing Brute-Force Attack on PRESENT Cipher

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

Martin Novotný
Faculty of Information Technology
Czech Technical University in Prague
Kolej 50/02, 160 00 Praha 6, Czech Republic
Email: novotmy@fel.cvut.cz

I. INTRODUCTION
Lightweight cryptography finds many applications in nowadays products. SmartCards, RFID tags or sensors require key system requirements. Unfortunately, many of such products suffer from cryptanalytically weak ciphers and/or implementations. Variety of attacks against ciphers like Mistar (Cryptol), KツνQ 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 higher cryptographic strength, while it enables space-saving hardware implementations. 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 Xeon processors X53551000 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 selected algorithm, the parallel hardware architecture can outperform conventional computers by several orders of magnitude. COPACOBANA has been designed under the assumption 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 do not require very local knowledge 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 group of a known plaintexts and a corresponding ciphertext. It is based on total 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 using the unverified keys and 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 coordinate a parallel attack. During initialization the values of plaintext and ciphertext are stored in each FPGA in corresponding registers. To parallelize the attack, the 80 bit key space is divided into 20 equal key subspaces defined by upper 39 bits of a key. During runtime, the host computer successively assigns each 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 success, the FPGA is assigned with another key subspace (seach in a 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 7.8% of chip area. The maximum achieved frequency is 120 MHz. Each FPGA can verify 2 keys per one clock cycle and the COPACOBANA is equipped with 100 FPGAs, the maximum performance is 2.88 Gbps (1st) or 2.74 Gbps (2nd) 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.