Studies in Computational Intelligence 296

Roger Lee (Ed.)

Guest Editors: Olga Ormandjieva • Alain Abran
Constantinos Constantinides

Software Engineering Research, Management and Applications 2010

Springer

Roger Lee (Ed.)

Software Engineering Research, Management and Applications 2010

Guest Editors

Olga Ormandjieva
Alain Abran
Constantinos Constantinides

Springer

Prof. Roger Lee
Software Engineering &
Information Technology Institute
Computer Science Department
Central Michigan University
Mt. Pleasant, MI 48859, U.S.A.
E-mail: lee1ry@cmich.edu

ISBN 978-3-642-13272-8 e-ISBN 978-3-642-13273-5

DOI 10.1007/978-3-642-13273-5

Studies in Computational Intelligence ISSN 1860-949X

Library of Congress Control Number: 2010927079

Typeset & Cover Design: Scientific Publishing Services Pvt. Ltd., Chennai, India.

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

Software Engineering Research, Management and Applications 2010

Studies in Computational Intelligence, Volume 296

Editor-in-Chief
Prof. Janusz Kacprzyk
Systems Research Institute
Polish Academy of Sciences
ul. Newelska 6
01-447 Warsaw
Poland
E-mail: kacprzyk@ibspan.waw.pl

Further volumes of this series can be found on our homepage: springer.com

Vol. 275. Dilip Kumar Pratihar and Lakhmi C. Jain (Eds.)
Intelligent Autonomous Systems, 2010
ISBN 978-3-642-11675-9

Vol. 276. Jacek Mańdziuk
Knowledge-Free and Learning-Based Methods in Intelligent Game Playing, 2010
ISBN 978-3-642-11677-3

Vol. 277. Filippo Spagnolo and Benedetto Di Paola (Eds.)
European and Chinese Cognitive Styles and their Impact on Teaching Mathematics, 2010
ISBN 978-3-642-11679-7

Vol. 278. Radomir S. Stankovic and Jaakko Astola
From Boolean Logic to Switching Circuits and Automata, 2010
ISBN 978-3-642-11681-0

Vol. 279. Manolis Wallace, Ioannis E. Anagnostopoulos, Phivos Mylonas, and Maria Bielikova (Eds.)
Semantics in Adaptive and Personalized Services, 2010
ISBN 978-3-642-11683-4

Vol. 280. Chang Wen Chen, Zhu Li, and Shiguo Lian (Eds.)
Intelligent Multimedia Communication: Techniques and Applications, 2010
ISBN 978-3-642-11685-8

Vol. 281. Robert Babuska and Frans C.A. Groen (Eds.)
Interactive Collaborative Information Systems, 2010
ISBN 978-3-642-11687-2

Vol. 282. Husrev Taha Sencar, Sergio Velastin, Nikolaos Nikolaidis, and Shiguo Lian (Eds.)
Intelligent Multimedia Analysis for Security Applications, 2010
ISBN 978-3-642-11754-1

Vol. 283. Ngoc Thanh Nguyen, Radoslaw Katarzyniak, and Shyi-Ming Chen (Eds.)
Advances in Intelligent Information and Database Systems, 2010
ISBN 978-3-642-12089-3

Vol. 284. Juan R. González, David Alejandro Pelta, Carlos Cruz, Germán Terrazas, and Natalio Krasnogor (Eds.)
Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), 2010
ISBN 978-3-642-12537-9

Vol. 285. Roberto Cipolla, Sebastiano Battiato, and Giovanni Maria Farinella (Eds.)
Computer Vision, 2010
ISBN 978-3-642-12847-9

Vol. 286. Zeev Volkovich, Alexander Bolshoy, Valery Kirzhner, and Zeev Barzily
Genome Clustering, 2010
ISBN 978-3-642-12951-3

Vol. 287. Dan Schonfeld, Caifeng Shan, Dacheng Tao, and Liang Wang (Eds.)
Video Search and Mining, 2010
ISBN 978-3-642-12899-8

Vol. 288. I-Hsien Ting, Hui-Ju Wu, Tien-Hwa Ho (Eds.)
Mining and Analyzing Social Networks, 2010
ISBN "Pending"

Vol. 289. Anne Håkansson, Ronald Hartung, and Ngoc Thanh Nguyen (Eds.)
Agent and Multi-agent Technology for Internet and Enterprise Systems, 2010
ISBN "Pending"

Vol. 290. Weiliang Xu and John Bronlund
Mastication Robots, 2010
ISBN "Pending"

Vol. 291. Shimon Whiteson
Adaptive Representations for Reinforcement Learning, 2010
ISBN "Pending"

Vol. 292. Fabrice Guillet, Gilbert Ritschard, Henri Briand, Djamel A. Zighed (Eds.)
Advances in Knowledge Discovery and Management, 2010
ISBN "Pending"

Vol. 293. Anthony Brabazon, Michael O'Neill, and Dietmar Maringer (Eds.)
Natural Computing in Computational Finance, 2010
ISBN "Pending"

Vol. 294. Manuel F.M. Barros, Jorge M.C. Guilherme, and Nuno C.G. Horta
Analog Circuits and Systems Optimization based on Evolutionary Computation Techniques, 2010
ISBN 978-3-642-12345-0

Vol. 295. Roger Lee (Ed.)
Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing 2010
ISBN 978-3-642-13264-3

Vol. 296. Roger Lee (Ed.)
Software Engineering Research, Management and Applications 2010
ISBN 978-3-642-13272-8

Preface

The purpose of the 8th Conference on Software Engineering Research, Management and Applications (SERA 2010) held on May 24 – 26, 2010 in Montreal, Canada was to bring together researchers and scientists, businessmen and entrepreneurs, teachers and students to discuss the numerous fields of computer science, and to share ideas and information in a meaningful way. Our conference officers selected the best 16 papers from those papers accepted for presentation at the conference in order to publish them in this volume. The papers were chosen based on review scores submitted by members of the program committee, and underwent further rounds of rigorous review.

In Chapter 1, Emil Vassev and Serguei Mokhov discuss their work in creating a Distributed Modular Audio Recognition Framework capable of self-healing using the Autonomic System Specification Language.

In Chapter 2, Yuhong Yan et al. present a new model of the Web Service Composition Problem and propose a reparative method based on planning graphs.

In Chapter 3, Chandan Sarkar et al. explore options for conducting remote usability tests using their newly-developed Total Cost of Administration (TCA) tool to collect and analyze test results.

In Chapter 4, Idir Ait-Sadoune and Yamine Ait-Ameur focus on the formal description, modeling, and validation of web services compositions and suggest a refinement based method that encodes the Business Process Execution Language (BPEL) model's decompositions.

In Chapter 5, Emil Vassev describes his work on code generation of autonomous systems using the Autonomic System Specification Language (ASSL) including an overview of ASSL and features of autonomously generated code.

In Chapter 6, Mohamed Miladi et al. propose a UML extension as a model based solution for the continuous increase in systems complexity and the necessity of cooperation between applications.

In Chapter 7, Haeng-Kon Kim and Roger Lee improve the efficiency of the proxy driving service with the addition of location-based service support.

In Chapter 8, Ahmad Hosseingholizadeh and Abdolreza Abhari propose a new approach to risk analysis by combining various metrics in an attempt to take into account all risky aspects of the software project.

In Chapter 9, Noorulain Khurshid et al. develop a categorical modeling language as the first step toward the creation of a powerful modeling paradigm capable of modeling emerging and evolving behavior of complex software.

In Chapter 10, Vieri Del Bianco et al. investigate the impact of economic factors on the adoption of Open Source Software.

In Chapter 11, Samir Ouchani et al. propose a verification methodology of a composition of UML behavioral diagrams.

In Chapter 12, Haeng-Kon Kim and Sun Myung Hwang address mobile application systems maintenance overhead with a knowledge discovery agent for an effective routing method using simple bit-map topology information.

In Chapter 13, D. Mouheb et al. present an aspect-oriented modeling approach for specifying and integrating security concerns into UML design models.

In Chapter 14, Eric Famutimi et al. present several techniques for using Python as a tool in computational analysis in one dimensional systems.

In Chapter 15, Fracisco Valdés Souto and Alain Abran describe an experiment conducted to compare the performance of the Estimation of Projects in Contexts of Uncertainty (EPCU) model against the Expert Judgement Estimation approach using data from industry projects.

In Chapter 16, Hossein Mehrfard et al. discuss the drawbacks of Extreme Programming (XP) when confronted with the stringent regulations for medical software imposed by the Food and Drug Administration (FDA) and propose an extension to XP to combat its weakness.

It is our sincere hope that this volume provides stimulation and inspiration, and that it will be used as a foundation for works yet to come.

May 2010

Roger Lee
Olga Ormandjieva
Alain Abran
Constantinos Constantinides

Contents

List of Contributors

Abdolreza Abhari
Ryerson University, ON, Canada
aabhari@ryerson.ca

Alain Abran
École de Technologie Supérieure
alain.abran@etsmtl.ca

Yamine Ait-Ameur
Laboratory of Applied Computer Science (LISI-ENSMA), France
yamine@ensma.fr

Othmane Ait Mohamed
Concordia University, QC, Canada
ait@ece.concordia.ca

Idir Ait-Sadoune
Laboratory of Applied Computer Science (LISI-ENSMA), France
idir.aitsadoune@ensma.fr

Eddy Bell
i-PEI LLC, WA, United States
eddy@i-pei.com

Vieri Del Bianco
University College Dublin, Ireland
viere.delbianco@ucd.ie

Mourad Debbabi
Concoria University, QC, Canada
debbabi@ece.concordia.ca

Khalil Drira
Université de Toulouse, France
khalil@laas.fr

Eric Famutimi
Central Michigan University, MI, United States
famut1eo@cmich.edu

Abdelwahab Hamou-Lhadj
Concordia University, QC, Canada
abdelw@ece.concordia.ca

Ahmad Hosseingholizadeh
Ryerson University, ON, Canada
ahossein@ryerson.ca

Sun Myung Hwang
Daejeon University, Korea
sunhwang@dju.ac.kr

Mohamed Jmaiel
University of Sfax, Tunisia
Mohamed.jmaiel@
enis.rnu.tn

Noorulain Khurshid
Concordia University, QC, Canada
N_khursh@
encs.concordia.ca

Haeng-Kon Kim
Catholic University of Daegu, Korea
hangkon@cu.ac.kr

Stan Klasa
Concordia University, QC, Canada
Klasa@
encs.concordia.ca

Dmitri Klementiev
i-PEI LLC, WA, United States
dklem@microsoft.com

Fatma Krichen
University of Sfax, Tunisia
Fatma.krichen@irit.fr

Luigi Lavazza
Università degli Studi dell' Insubria, Italy
luigi.lavazza@uninsubria.it

Roger Lee
Central Michigan University, MI, United States
lee1ry@cmich.edu

V. Lima
Concordia University, QC, Canada
v_nune@ciise.concordia.ca

Hossein Mehrfard
Concordia University, QC, Canada
H_mehrfa@ece.concordia.ca

Mohamed Nadhmi Miladi
University of Sfax, Tunisia
Mohammednadhmi.miladi@isimsf.rnu.tn

Serguei Mokhov
Concordia University, QC, Canada
mokhov@cse.concordia.ca

Sandro Morasca
Università degli Studi dell'Insubria, Italy
sandro.morasca@uninsubria.it

D. Mouheb
Concordia University, QC, Canada
d_mouheb@ciise.concordia.ca

M. Nouh
Concordia University, QC, Canada
m_nouh@ciise.concordia.ca

Olga Ormandjieva
Concordia University, QC, Canada
ormandj@encs.concordia.ca

Samir Ouchani
Concordia University, QC, Canada
s_oucha@ece.concordia.ca

Heidar Pirzadeh
Concordia University, QC, Canada
s_pirzad@ece.concordia.ca

Pascal Poizat
University of Evry Val d'Essonne, France
cal.poizat@lri.fr

Makan Pourzandi
Ericsson Canada Inc., QC, Canada
pourzandi@ericsson.com

Chandan Sarkar
Michigan State University, MI, United States
sarkarch@msu.edu

Michael Stinson
Central Michigan University, MI, United States
stins1m@cmich.edu

Candace Soderston
Microsoft Corporation, WA, United States
csoders@microsoft.com

Francisco Valdés Souto
École de Technologie Supérieure
francisco.valdes@spingere.com.mx

Davide Taibi
Università degli Studi dell'Insubria, Italy
Davide.taibi@uninsubria.it

C. Talhi
Concordia University, QC, Canada
talhi@ciise.concordia.ca

Davide Tosi
Università degli Studi dell'Insubria, Italy
davide.tosi@uninsubria.it

Emil Vassev
University College Dublin, Ireland
emil-vassev@lero.ie

L. Wang
Concordia University, QC, Canada
wang@ciise.concordia.ca

Yuhong Yan
Concordia University, QC, Canada
yuhong@cse.concordia.ca

Ludeng Zhao
Concordia University, QC, Canada
ludeng.zhao@encs.concordia.ca

Towards Autonomic Specification of Distributed MARF with ASSL: Self-healing

Emil Vassev and Serguei A. Mokhov

Abstract. In this paper, we discuss our work towards self-healing property specification of an autonomic behavior in the Distributed Modular Audio Recognition Framework (DMARF) by using the Autonomic System Specification Language (ASSL). ASSL aids in enhancing DMARF with an autonomic middleware that enables it to perform in autonomous systems that theoretically require less-to-none human intervention. Here, we add an autonomic middleware layer to DMARF by specifying the core four stages of the DMARF's pattern-recognition pipeline as autonomic elements managed by a distinct autonomic manager. We devise the algorithms corresponding to this specification.

1 Introduction

The vision and metaphor of autonomic computing (AC) [7] is to apply the principles of self-regulation and complexity hiding. The AC paradigm emphasizes on reducing the workload needed to maintain complex systems by transforming them into self-managing autonomic systems. The idea is that software systems can manage themselves, and deal with on-the-fly occurring requirements automatically. This is the main reason why a great deal of research effort is devoted to the design and development of robust AC tools. Such a tool is the ASSL (Autonomic System Specification Language) framework, which helps AC researchers with problem formation,

Emil Vassev
Lero-the Irish Software Engineering Research Centre, University College Dublin, Dublin, Ireland
e-mail: emil.vassev@lero.ie

Serguei A. Mokhov
Department of Computer Science and Software Engineering, Concordia University, Montreal, QC, Canada
e-mail: mokhov@cse.concordia.ca

Roger Lee (Ed.): SERA 2010, SCI 296, pp. 1–15, 2010.
springerlink.com

system design, system analysis and evaluation, and system implementation. In this work, we use ASSL [16, 15] to integrate autonomic features into the Distributed Modular Audio Recognition Framework (DMARF) – an intrinsically complex system composed of multi-level operational layers.

Problem Statement and Proposed Solution

Distributed MARF – DMARF – could not be used in autonomous systems of any kind "as-is" due to lack of provision for such a use by applications that necessitate autonomic requirements. Extending DMARF directly to support the said requirements is a major design and development effort for an open-source project.

In this work, we provide a proof-of-concept ASSL specification of one of the three core autonomic requirements for DMARF – self-healing (while the other two – self-protection and self-optimization are done in the work). In Appendix is the current outcome for the self-healing aspect and the rest of paper describes the methodology and related work behind it. Having the ASSL specification completed allows compiling it into the wrapper Java code as well as management Java code to provide an autonomic layer to DMARF to fulfill the autonomic requirement.

The rest of this paper is organized as follows. In Section 2, we review related work on AS specification and code generation. As a background to the remaining sections, Section 3 provides a brief description of both DMARF and ASSL frameworks. Section 4 presents the ASSL self-healing specification model for DMARF. Finally, Section 5 presents some concluding remarks and future work.

2 Related Work

IBM Research has developed a framework called Policy Management for Autonomic Computing (PMAC) [1], which provides a standard model for the definition of policies and an environment for the development of software objects that hold and evaluate policies. PMAC is used for the development and management of intelligent autonomic software agents. With PMAC, these agents have the ability to dynamically change their behavior, ability provided through a formal specification of policies encompassing the scope under which these policies are applicable. Moreover, policy specification includes the conditions under which a policy is in conformity (or has been violated), a set of resulting actions, goals or decisions that need to be taken and the ability to determine the relative value (priority) of multiple applicable actions, goals or decisions. [1]

3 Background

In this section, we introduce both the DMARF and the ASSL frameworks, thus needed to understand the specification models presented in Section 4.

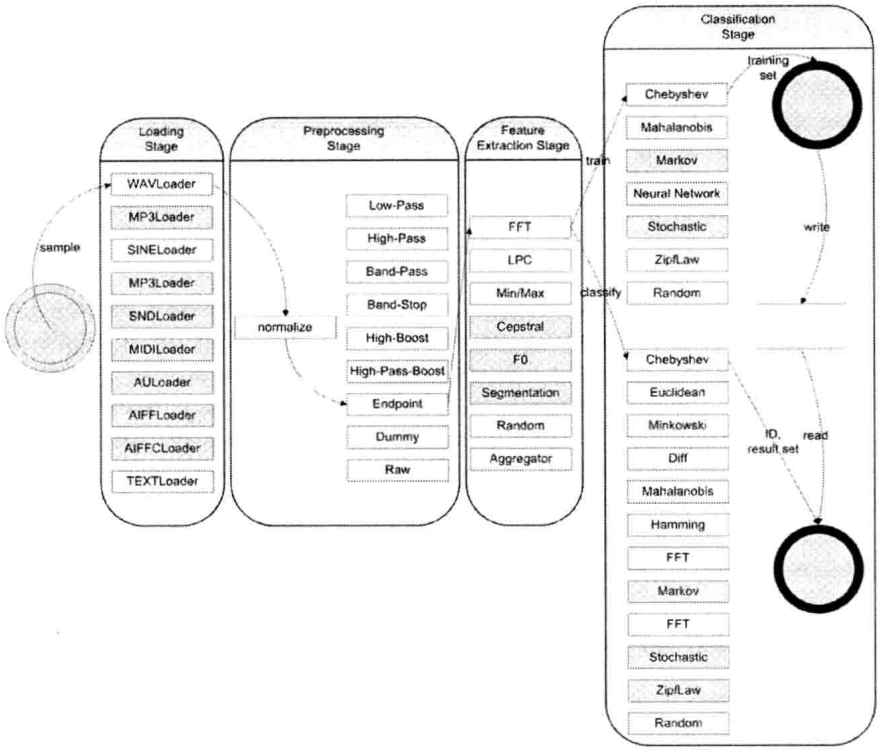

Fig. 1 MARF's Pattern Recognition Pipeline

3.1 Distributed MARF

DMARF [2, 5] is based on the classical MARF whose pipeline stages were made into distributed nodes.

Classical MARF

The Modular Audio Recognition Framework (MARF) [3] is an open-source research platform and a collection of pattern recognition, signal processing, and natural language processing (NLP) algorithms written in Java and put together into a modular and extensible framework. MARF can run distributively over the network, run stand-alone, or just act as a library in applications. The backbone of MARF consists of pipeline stages that communicate with each other to get the data they need in a chained manner. In general, MARF's pipeline of algorithm implementations is presented in Figure 1, where the implemented algorithms are in white boxes. The pipeline consists of four basic stages: sample loading, preprocessing, feature extraction, and training/classification. There are a number of applications that test MARF's functionality and serve as examples of how to use MARF's modules. One of the most prominent applications `SpeakerIdentApp` – Text-Independent Speaker Identification (who, gender, accent, spoken language, etc.).

Distributed Version

The classical MARF was extended [5, 4] to allow the stages of the pipeline to run as distributed nodes as well as a front-end, as roughly shown in Figure 2. The basic stages and the front-end perform communication over Java RMI [17], CORBA [8], and XML-RPC WebServices [9].

Applications of DMARF

High-volume processing of recorded audio, textual, or imagery data are possible pattern-recognition and biometric applications of DMARF. Most of the emphasis in this work is in audio, such as conference recordings with purpose of attribution of said material to identities of speakers. Similarly, processing a bulk of recorded phone conversations in a police department for forensic analysis and subject identification and classification, where sequential runs of the MARF instances on the same machine, especially a mobile equipment such as a laptop, PDA, cellphone, etc. which are not high-performance, an investigator has an ability of uploading collected voice samples to the servers constituting a DMARF-implementing network.

3.1.1 DMARF Self-healing Requirements

DMARF's capture as an autonomic system primarily covers the autonomic functioning of the distributed pattern-recognition pipeline. We examine properties that apply to DMARF and specify in detail the self-healing aspect of it.

If we look at the pipeline as a whole, we see that there should be at least one instance of the every stage somewhere on the network. There are four main core pipeline stages and application-specific stage that initiates pipeline processing. If one of the core stages goes offline completely, the pipeline stalls, so to recover one needs a replacement node, or recovery of the failed node, or to reroute the pipeline

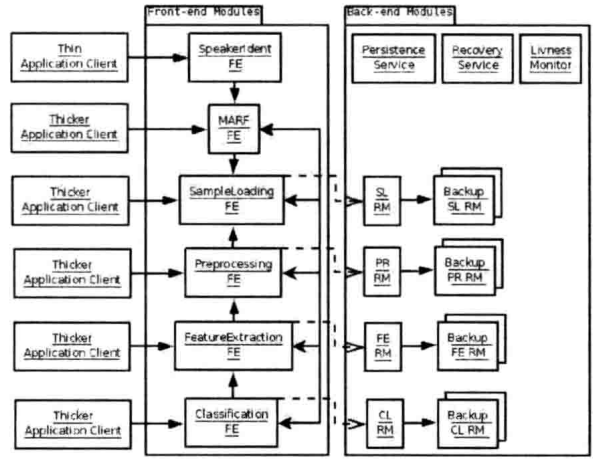

Fig. 2 The Distributed MARF Pipeline

through a different node with the same service functionality the failed one used to provide until the failed one recovers so a pipeline has to always self-heal and provide at least one pipeline route to be usable.

- A DMARF-based system should be able to recover itself in the form of replication to keep at least one route of the pipeline available. There are two types of replication – the replication of a service, which essentially means we increase the number of nodes per core stage (e.g. two different hosts provide preprocessing services as in active replication, so if one goes down, the pipeline is still not stalled; if both are up they can contribute to load balancing which is a part of the self-optimization aspect) and replication within the node itself. The latter replicas do not participate in the computation. They only receive updates and are on stand-by if the primary service goes down.
- If all nodes of a core stages go down, the stage preceding it is responsible to start up a temporary one on the host of the preceding stage, set it up to repair the pipeline. This is the hard replication needed to withstand stall faults, where it is more vulnerable and not fault-tolerant.
- In the second case, denoting passive replication of the same node (or even different nodes) losing a primary or a replica is not as serious as in the first case because such a loss does not produce a pipeline stall and it is easier to self-heal after a passive replica loss.
- Restart and recovery of the actual failed node without replicas is another possibility for self-healing for DMARF. Technically, it may be tried prior or after the replica kicks in.

3.2 ASSL

The Autonomic System Specification Language (ASSL) [15, 16] approaches the problem of formal specification and code generation of ASs within a framework. The core of this framework is a special formal notation and a toolset including tools that allow ASSL specifications be edited and validated. The current validation approach in ASSL is a form of consistency checking (handles syntax and consistency errors) performed against a set of semantic definitions. The latter form a theory that aids in the construction of correct AS specifications. Moreover, from any valid specification, ASSL can generate an operational Java application skeleton.

In general, ASSL considers autonomic system (ASs) as composed of autonomic elements (AEs) interacting over interaction protocols. To specify those, ASSL is defined through formalization tiers. Over these tiers, ASSL provides a multi-tier specification model that is designed to be scalable and exposes a judicious selection and configuration of infrastructure elements and mechanisms needed by an AS. The ASSL tiers and their sub-tiers (cf. Figure 3) are abstractions of different aspects of the AS under consideration. They aid not only to specifying the system at different levels of abstraction, but also to reducing the complexity, and thus, to improving the overall perception of the system.