RECENT ADVANCES in SOFTWARE ENGINEERING, PARALLEL and DISTRIBUTED SYSTEMS

Editors:
Prof. Leon Trilling, Massachusetts Institute of Technology (MIT), USA
Prof. D. Perkins, Harvard University, USA
Prof. Dionysios (Dion) D. Dionysiou, University of Cincinnati, USA
Prof. Leonid Perlovsky, Harvard University, USA
Prof. Kent Davey, IEEE Fellow, Editor IEEE Trans. on Magnetics, Austin, TX, USA
Prof. David Landgrebe, Purdue University, USA
Prof. Miguel A. Marino, Distinguished Professor of Hydrology, Civil & Environmental Eng., and Biological & Agricultural Engineering, University of California, CA, USA
Prof. D. L. Russell, Professor of Mathematics, Virginia Tech, USA
Prof. Steven H. Collicott, School of Aeronautics and Astronautics, Univ. West Lafayette,USA
Prof. Marco Ceccarelli, (IFToMM President elect 2008-2011), University of Cassino, IT
Prof. John W. Lund, PE, Professor Emeritus of Civil Engineering,
Past President of the Intern. Geothermal Association, Oregon Institute of Technology, USA

RECENT ADVANCES in SOFTWARE ENGINEERING, PARALLEL and DISTRIBUTED SYSTEMS

Proceedings of the 8th WSEAS International Conference on
SOFTWARE ENGINEERING, PARALLEL and DISTRIBUTED SYSTEMS (SEPADS '09)

Cambridge, UK, February 21-23, 2009

Electrical and Computer Engineering Series
A Series of Reference Books and Textbooks

ISBN: 978-960-474-052-9
ISSN: 1790-5117

Published by WSEAS Press
www.wseas.org

RECENT ADVANCES in SOFTWARE ENGINEERING, PARALLEL and DISTRIBUTED SYSTEMS

Proceedings of the 8th WSEAS International Conference on SOFTWARE ENGINEERING, PARALLEL and DISTRIBUTED SYSTEMS (SEPADS '09)

Cambridge, UK
February 21-23, 2009

Electrical and Computer Engineering Series
A Series of Reference Books and Textbooks

Published by WSEAS Press
www.wseas.org

ISSN: 1790-5117
ISBN: 978-960-474-052-9

RECENT ADVANCES in SOFTWARE ENGINEERING, PARALLEL and DISTRIBUTED SYSTEMS

Proceedings of the 8th WSEAS International Conference on SOFTWARE ENGINEERING, PARALLEL and DISTRIBUTED SYSTEMS (SEPADS '09)

Cambridge, UK
February 21-23, 2009

Electrical and Computer Engineering Series
A Series of Reference Books and Textbooks

Published by WSEAS Press
www.wseas.org

Copyright © 2009, by WSEAS Press

All the copyright of the present book belongs to the World Scientific and Engineering Academy and Society Press. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the Editor of World Scientific and Engineering Academy and Society Press.

All papers of the present volume were peer reviewed by two independent reviewers. Acceptance was granted when both reviewers' recommendations were positive.
See also: http://www.worldses.org/review/index.html

ISSN: 1790-5117
ISBN: 978-960-474-052-9

World Scientific and Engineering Academy and Society

Table of Contents

Plenary Lecture 1: Software Risk Management through a Shared Approach 9
Khairuddin Hashim

Plenary Lecture 2: Measuring Software Product Quality with ISO Standards: An Information Model 10
Rafa E. Al-Qutaish

Plenary Lecture 3: The Efficiency of Parallel Metaheuristics for Combinatorial Optimization – Paradigms, Models and Implementations 11
Plamenka Borovska

Real Time Earthquake Azimuth Estimation using Parallel Wavelet Transform 13
G. Hloupis, F. Vallianatos, J. P. Makris

How Satisfactory is it to Communicate Customer Knowledge with the Aid of Multimodal Interaction Metaphors? 21
Mutlaq B. Alotaibi, Dimitrios I. Rigas

Simulation Study of the Functioning of LFSR for Grade 4 Irreducible Polynomials 27
Mirella Amelia Mioc

An Approach to Derive the Use Case Diagrams from an Event Table 33
Mohammad I. Muhairat, Rafa E. Al-Qutaish

The Analytical Hierarchy Process as a Tool to Select Open Source Software 39
Rafa E. Al-Qutaish, Mohammad I. Muhairat, Basil M. Al-Kasasbeh

PVS Metamodel 45
Lukman Ab. Rahim

Quality Product Derivation: A Case Study for Quality Control at Siemens 51
Clotilde Rohleder

Ball Nose Milling Cutter Radius Compensation in Z Axis for CNC 57
Dragoi Mircea Viorel

Flexible Superfinishing Modules used in SME (Small and Medium Enterprises) 61
Badea Lepadatescu, Adriana Fota, Anisor Nedelcu, Constantin Buzatu, Adela-Eliza Dumitrascu, Ioan Enescu

Measuring Communication Gap in Software Requirements Elicitation Process 66
Abdullah Mohd Zin, Noraini Che Pa

Theoretical Validation of Object-Oriented Lack-of-Cohesion Metrics 72
Jehad Al Dallal

A Compact CPN representation for Embedded and Control Systems Fault Diagnosis and Recovery 78
Anthony Spiteri Staines

A Group Synchronization Algorithm for VoIP Conferencing 84
A. Dutta, R. Dasgupta, S. Bhattacharya

BONDIN: A New Engineering Simulation Software for ODE and DAE Systems with Symbolic Notation based in the Bond Graph Technique 90
G. Romero, J. Felez, J. M. Cabanellas, J. Maroto

An Architecture for Consumer Support Systems 98
Jyhjong Lin

An Adaptive Fault Detection Scheme forWireless Sensor Networks 106
Jae-Young Choi, Sung-Jib Yim, Yoon Jae Huh, Yoon-Hwa Choi

A Partitional Clustering Algorithm for Crosscutting Concerns Identification 111
Gabriela Czibula, Grigoreta Sofia Cojocar, Istvan Gergely Czibula

A Framework for Software Requirements Engineering 117
Khairuddin Hashim, Nurul Naslia Khairuddin

Users' Views of Facial Expressions and Body Gestures in E-Learning Interfaces: An Empirical Evaluation 121
Marwan Alseid, Dimitrios Rigas

The Use of Multimodality Metaphors in E-learning 127
Dimitrios Rigas, Mohamed Sallam

Software Engineering Assessments and Learning Outcomes 131
Khairuddin Hashim, Nurul Naslia Khairuddin

Modeling Service-Driven Architecture for Distributed Enterprise Portal for Maintaining Global Patch Consistency 135
Hui-Ling Lin, Shao-Shin Hung, Derchian Tsaih

An OCL Extension for Checking and Transforming UML Models 144
Thierry Millan, Laurent Sabatier, Thanh-Thanh Le Thi, Pierre Bazex, Christian Percebois

Comparison of I/O Scheduling Algorithms for High Parallelism MEMS-Based Storage Devices 150
Eunji Lee, Kern Koh, Hyunkyoung Choi, Hyokyung Bahn

Analyses of Task Based Learning in Developing "M-Learn" Mobile Learning Software Solution: Case Study 156
Majlinda Fetaji, Bekim Fetaji

The Fourth Dimension of Information System Audit and Security 162
Akshai Aggarwal, Sujata Kanhere, Vishnu Kanhere, Shankar Kanhere

A New Fuzzy TOPSIS Method for Material Handling System Selection Problems 169
Mehdi Amiri Aref, Nikbakhsh Javadian

Authors Index 175

An Approach to Derive the Use Case Diagrams from an Event Table

Mohammad I. Muhairat and Rafa E. Al-Qutaish
Department of Software Engineering
Al-Zaytoonah University of Jordan
Airport Street, P.O. Box: 130, Amman 11733
JORDAN
mohmuhaba14@yahoo.com, rafa@ieee.org

Abstract:- Building the use-case diagram is a very important task since it represents a transition between the requirements and design phases. However, building such diagram is a time consuming process and needs a complete understanding of the requirements. In this paper, we introduce an approach to derive use case diagrams from an event table. This new approach will facilitate and speed the generation process of the use case diagrams. However, this approach will completely depends on the availability of a comprehensive event table which to be built from the available requirements.

Keywords:- Software Requirements, Software Design, Event Table, Use Case Diagram.

1 Introduction

Nowadays, there are many different approaches in the software engineering literature to identify the use cases, for examples:

1. Listing all users and define there needs [1], [2], [3], [4], [5];
2. Defining all system functions and adding new functions that user may be need [1], [2], [3], [4], [5];
3. List all graphical user interfaces [2],[6];
4. Defining all users' goals of system [1], [2], [7], [8], [9].

In industry, many practitioners are using the third approach to get an initial list of use cases. Furthermore, the event decomposition technique [1], [2], [5] is the most used one for defining a use case model. This technique focusing on the events a system must respond to and looking at how a system responds.

An event is an occurrence at specific time and place, can be described, and should be remembered by the system [2], [11] [16].

Building the use-case diagram is a very important task since it represents a transition between the requirements and design phases. However, building such diagram is a time consuming process and needs a complete understanding of the requirements. In this paper, we introduce an approach to derive the use case diagrams from an event table. This new approach will facilitate and speed the generation process of the use case diagrams. However, this approach will completely depends on the availability of a comprehensive event table which to be built from the available requirements.

The rest of this paper is organized as follows: Section 2 presents a general overview on the related concepts, that is, on the event table and use case concepts. In Section 3, the approach of deriving the use case diagram from the event table has been explained. Section 4 gives an example of the implementation of the proposed approach. Finally, Section 5 concludes the paper and introduces the potential future works.

2 Related Concepts

2.1 Event Table

Since 1980's, the event analysis technique [10] [11] [12] has been the preferred one for the practitioners. The results of event analysis are documented in an event table. In the structured approach, event analysis recognizes a basic set of processes. Whereas, in the object-oriented analysis, each event discourse an essential use case [13]. Furthermore, an event table can be created from the external events to support the use case diagrams [14]. In addition, the event table has been used by Gargantini and Heitmeyer [15] to generate a suite test sequences.

In the event table, there are three types of events, that is:

1. *External Event (EE)*: an event that occurs outside of the system, usually initiated by external actor or user; for example: *student wants to search for a book item*.

2. *Temporal Internal Event (TIE)*: an event that happens when the system reaches a specific point of time; for example: *time to print book items report*.
3. *Conditional Internal Event (CIE)*: an event occurs when something happens inside the system and the system must initiate some process to response for this event; for example: *student reordered new books when the reordered point is reached*.

Business modeling help analyst to understand the business process. As result of that modeling, business events are identified and documented in an event table. Event table is a list of actions that lists events in rows and the information about each event in columns. Analyst can use event table to define use case model. However, analyst has to make some decisions when building a use case model.

The first one is to combine business events into one use case. For example, the business events of adding, deleting, updating customer information, analyst can combined them in one use case- Maintain Customer Information. Also, analyst makes decisions to split one business event into multiple use cases. For example, the business event customer withdraws his cash, analyst can split it two use cases Withdraw Cash and Identify Customer and can identify the relationship between them, for example: *Withdraw Cash <<include>> Identify Customer*.

From the above example (*student wants to search for a book item*) and for an EE, we can split this event into three main parts:

EE
- Student- Source
- Search- Action
- Book item-affected object

In addition, the TIE example (*time to print book items report*) can be divided into three main parts as the following:

TIE
- Time or "System"- Source
- Print- Action
- Book item- Affected object

Finally, we can split the CIE (*student reordered new books when the reordered point is reached*) into the following three main parts:

CIE
- Student - Source
- Reordered- Action "if the reordered point is reached"
- Book item- Affected object

As a result of this splitting, analyst defined the core elements "columns" for the proposed event table as in Table 1.

Table 1: Event Table.

Event	Source	Action	Object	Destination

Where:

Event: an event which causes the system to do something.

Source: the source of an event (an actor for an EE and the system for a TIE and CIE).

Action: the system functionality which we need.

Object: The object affected by this action.

Destination: An actor that receive the result of an event execution

For the purpose of building a complete use case model, we can extend this table and make some modification to contain other elements, see Table 2.

Table 2: Extended Event Table.

Event	General Source	Special Source	Action	Object	Includes "Action"	Extends "Action"	Specializes "Action"	Destination

Where:

General Source or Special Source: The type of an event source. We used these columns to define a Generalization / Specialization relationship between sources.

Includes: We used this column to determine the existence of includes relationship between actions.

Extends: We used this column to determine the existence of extends relationship between actions.

Specializes: We used this column to determine the existence of specializes relationship between actions.

2.2 Use Case Diagram

Use case diagram captures the system or subsystem behaviour. It represents the interaction between the actors and pieces of functionality called use cases. An actor is an idealization of a role played by an external person, process, or thing interacting with the system, subsystem, or class [7], [18]. Actors participate with one or more use cases by exchanging messages.

Actors may be defined in generalization hierarchies, in which an abstract actor description is shared and augmented by one or more specific actor descriptions. An actor may be a human, a computer system, or some executable process. An actor is drawn as a small stick person with the name below it. A use case is a coherent unit of externally visible functionality provided by a classifier (called the subject) and expressed by sequences of messages exchanged by the subject and one or more actors of the system unit [17], [18]. A use case can participate in several relationships; in addition to association with actors (see Table 3).

Recently, Snoeck [16] has proposed a new form of event table which is called object-event table to be a useful technique for modelling interaction between domain object types.

Table 3: Use Case Relationships Types.

Relationship	Function	Notation
Association	To indicate the communication between actors and uses cases.	————
Extend	To indicate the insertion of additional behavior into a base use case.	- - - - - - - - - - - - - - -> << extend>>
Include	To describes a behavior that is inserted explicitly into a base use case.	- - - - - - - - - - - - - - -> <<include>>
Use case or actor generalization	To indicate the communication between a general use case (actor) and a more specific use case (actor) that inherits and adds features to it.	——▷

3 Deriving the Use Case Diagram

This process consists of six steps, to build and check the correctness of the use case diagram.

1. Identify the actors for each event or action from the sources and destinations.
2. Identify the relationships between actors, if exists. There is only one type of relationship between actors, that is, a generalization / specialization relationship.
3. Identify the use cases. The analyst can derive the use case from the action which proceeded by an actor.
4. Identify the relationships between use cases, if exists. As we mentioned above, there is three type of relationships between use cases (includes, extends, and specializes).
5. Integrate all use cases and actors with all relationships types in one use case diagram.
6. Test the use case diagram.

From a given event table we can map the use cases and the actors, as in Figure 1.

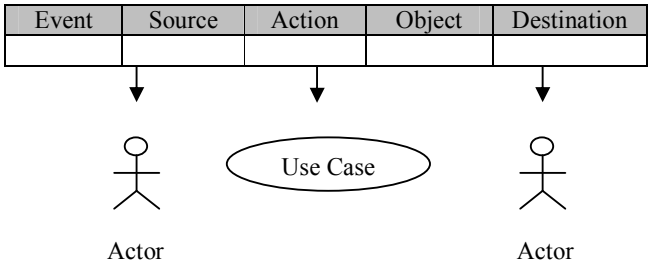

Figure 1: Mapping the Use Cases and Actors from an Event Table.

However, this mapping could be implemented using the two types of events, that is, external event (EE), and internal events (TIE, CIE). Figures 2 and 3 illustrate examples of both types, respectively.

Event	Source	Action	Object	Destination
Student wants to search for a book item	Student	Search for a book item	Book item	Student

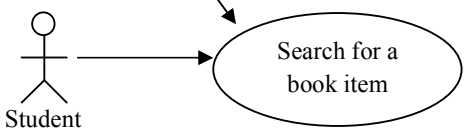

Figure 2: An Example of Mapping a Use Case and Actor from an EE.

Event	Source	Action	Object	Destination
Time to print a book items report	System	print a book items report	book items	Library Employee

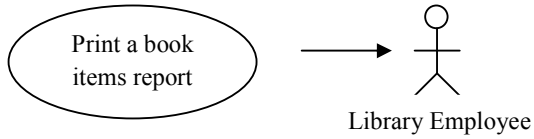

Figure 3: An Example of Mapping a Use Case and Actor from a TIE and CIE.

5 An Example

To demonstrate the feasibility of our approach, this approach has been applied for a case study. This section will implement the process of deriving a use case model from an event table for car rental web site, see Table 4 for its event table.

Table 4: The Event Table of the Car Rental Web Page.

Event	General Source	Special Source	Action	Object	Includes "Action"	Extends "Action"	Specializes "Action"	Destination
Customer browses the car models index	Customer	Member Non Member	Browse the car models index	Car model	View results	-	Look for a car model	Customer
Customer searches for a car model	Customer	Member Non Member	searches for a car model	Car model	View results	-	Look for a car model	Customer
Customer views the results	Customer	Member Non Member	view results	Car model	-		-	Customer
member log on to the web site	Member	-	Log on to the web site	Web page	-		-	Member
member view his details	Member	-	View member details	Member	-	member log on to the web site	-	Member
member log off from the web site	Member	-	Log off from the web site	Web page	-	member log on to the web site	-	Member
Member makes a reservation	Member	-	Make reservation	Reservation	-	member log on to the web site	-	Member
Member views a reservation	Member	-	view reservation	Reservation	-	member log on to the web site	-	Member

The following are the steps which will be used to build the use case diagram from the given event table (Table 4) for our example:

1. From the sources and destinations fields, we can define three actors: *customer*, *member* and *non member*.
2. From the general and special sources fields, we can define a *generalization / specialization* relationship between actors.
3. From the actions field, we can derive all the *use cases*, as the following:
   - *Browse a car models index*,
   - *Searches for a car models*,
   - *View results*,
   - *Logs on the web page*,
   - *View member details*, and
   - *Logs off the web page*.
4. From the "includes action", extends action, and specialize action fields, we can define all the relationship types between the use cases to be *includes*, *extends* and *specialize*. In addition, a new use case can be defined from specializes action field.
5. Now, we will integrate the all elements to build the use case diagram, as in Figure 4.

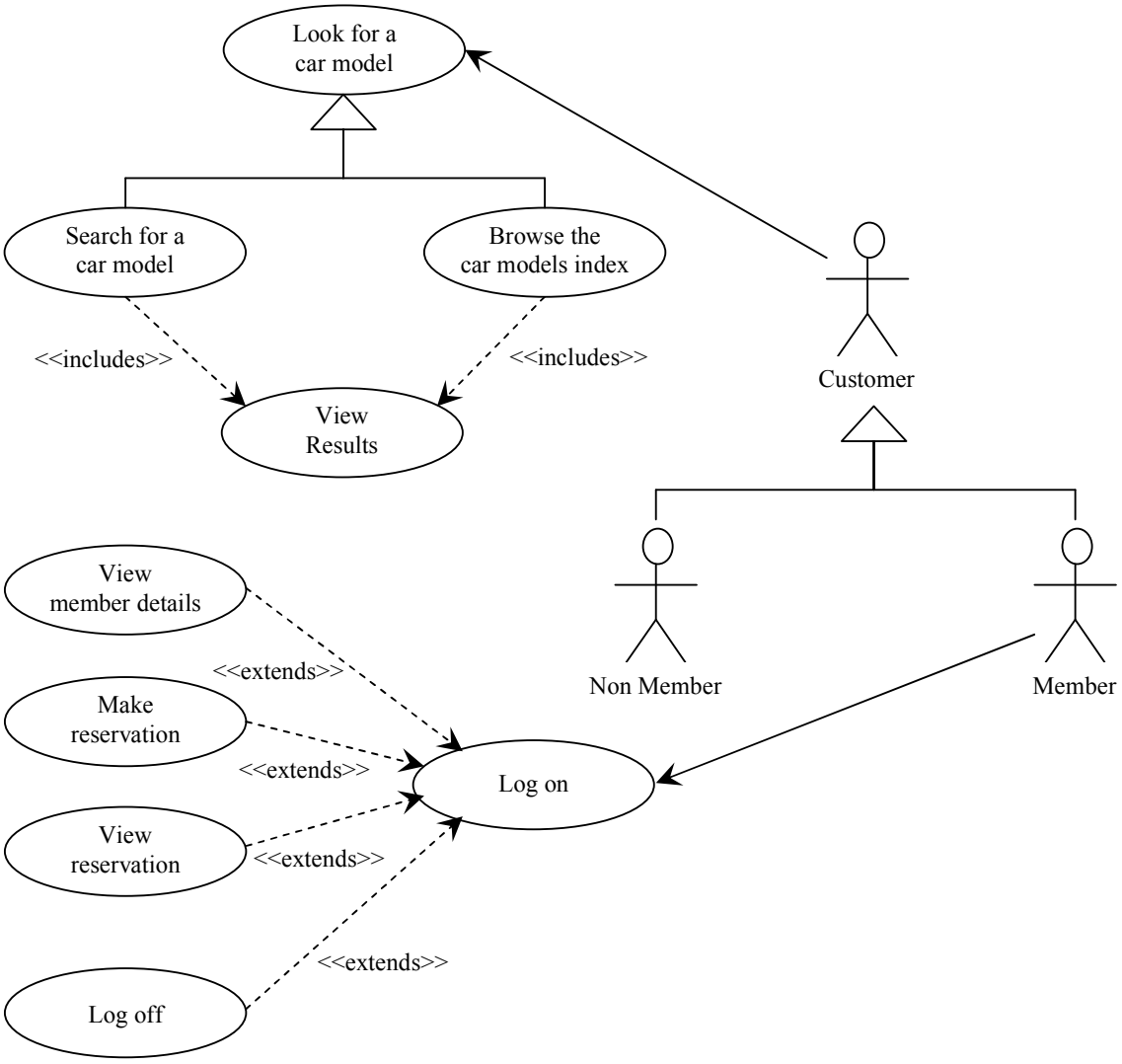

Figure 4: The Integrated Use Case Model.

6. Finally, the following steps could be used to test the use case diagram and to test the use cases against functional requirements:
   a. Check if all events are covered by actions "use cases;"
   b. Check if each use case represents a functional requirement;
   c. Check if all actors are well defined;
   d. Check if the relationships between the actors are defined;

e. Check if the relationships between the use cases are defined; and
f. Check the relationship between the different actors and use cases.

6 Conclusions and Future Work

Constructing the use-case diagram is a very important and essential task to go ahead to the design process. However, the use case diagram represents a transition stage between the requirements and design phases. Furthermore, building such diagram is a time consuming task and needs a complete understanding of the user requirements. In this paper, we have introduced an approach to derive the use case diagram from an event table. This new approach will facilitate and speed up the generation process of the use case diagrams. However, this approach is completely depending on the availability of a comprehensive event table which should be built during earlier tasks from the available user requirements.

It can be clearly noted from the above sections that this approach gives an ideal and reasonable methodology to build the intended use case diagram from any comprehensive event table. In addition, this approach will save the time for the building process of the use case diagram.

As a future work, this approach could be automated by developing a CASE tool to take the event table as an input and produce the intended use case diagram as an output.

References

[1] J. W. Satzinger, R. B. Jackson and S. D. Burd, Object-Oriented Analysis and Design with the Unified Process, 4th ed., Thomson Course Technology, 2005.

[2] C. Larman, *Applying UML and patterns: An Introduction to Object Oriented Analysis and Design and Iterative Development*, 3rd ed., Prentice Hall, USA, 2005.

[3] S. Bennett, S. McRobb and R. Farmer, *Object-Oriented Systems Analysis and Design Using UML*, McGraw Hill Education, USA, 2005.

[4] S. R. Schach, *An Introduction to Object-Oriented System Analysis and Design with UML and Unified Process*, McGraw-Hill, USA, 2003.

[5] P. R. Reed, *Developing Applications with Java and UML*, Addison Wesley, Boston, MA, USA, 2001.

[6] A. Cockburn, *Writing Effective Use Cases*, Addison-Wesley, Boston, MA, USA, 2001.

[7] Y. Liang, From Use Cases to Classes: a Way of Building Object Model with UML, *Journal of Information and Software Technology*, Vol. 45, 2003, pp. 83-93.

[8] L. Chung and S. Supakkul, Representing NFRs and FRs: A Goal-Oriented and Use Case Driven Approach, in *Proceedings of the 2nd International Conference on Software Engineering Research, Management and Applications (SERA'04)*, Los Angeles, CA, USA, pp. 29–41, 2005

[9] J. Lee and N. Xue, Analyzing user requirements by use cases: A goal-driven approach, *IEEE Software*, Vol. 16, No. 4, pp. 92–100, 1999.

[10] S. M. McMenamin and J. F. Palmer, *Essential Systems Analysis*, Yourdon Press, NY, USA, 1985.

[11] M. Page-Jones, *Fundamentals of Object-Oriented Design in UML*, Addision-Wesley, Boston, MA, USA, 1988.

[12] Yourdon, E., *Modern Structured Analysis*, Yourdon Press, Englewood Hills, NJ, USA, 1989.

[13] R. Stumpf and L. Teague, "Teachings Object-Oriented System Analysis and design with UML", in *Proceedings of the Information Systems Education (ISECON'05)*, Columbus, OH, USA, 2005.

[14] A. Purhonen, *Quality Driven Multimedia DSP Software Architecture Development*, Julkaisija-Utgivare Publisher, Oulu, Finland, 2002.

[15] A. Gargantini and C. Heitmeyer, "Using Model Checking to Generate Tests from Requirements Specifications", in *Proceedings of the European Software Engineering Conference (ESEC'99)*, 1999, pp. 146-162.

[16] M. Snoeck, G. Dedene, "Core Modelling Concepts in Object-Oriented conceptual Modeling", in *Proceedings of the Technology of Object-Oriented Languages and Systems Conference*, 2001, pp.170-179.

[17] W. Boggs and M. Boggs, *Mastering UML with Rational Rose*, SYBEX Inc., 2002.

[18] J. Rumbaugh, I. Jacobson and G. Booch, *The Unified Modeling Language Reference Manual*, 2nd ed., Addison-Wesley, Boston, MA, USA, 2004.