Advances in Intelligent and Soft Computing

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. 87. E. Corchado, V. Snášel, J. Sedano, A.E. Hassanien, J.L. Calvo, and D. Śliżak (Eds.)
Soft Computing Models in Industrial and Environmental Applications,
6th International Workshop SOCO 2011
ISBN 978-3-642-19643-0

Vol. 88. Y. Demazeau, M. Pěchoucěk, J.M. Corchado, and J.B. Pérez (Eds.)
Advances on Practical Applications of Agents and Multiagent Systems, 2011
ISBN 978-3-642-19874-8

Highlights in Practical Applications of Agents and Multiagent Systems, 2011
ISBN 978-3-642-19916-5

Vol. 90. J.M. Corchado, J.B. Pérez, K. Hallenborg, P. Golinska, and R. Corchuelo (Eds.)
Trends in Practical Applications of Agents and Multiagent Systems, 2011
ISBN 978-3-642-19930-1

Vol. 91. A. Abraham, J.M. Corchado, S.R. González, J.F. de Paz Santana (Eds.)
International Symposium on Distributed Computing and Artificial Intelligence, 2011
ISBN 978-3-642-19933-2

Vol. 92. P. Novais, D. Preuveneers, and J.M. Corchado (Eds.)
Ambient Intelligence - Software and Applications, 2011
ISBN 978-3-642-19936-3

5th International Conference on Practical Applications of Computational Biology & Bioinformatics 6-8th, 2011
ISBN 978-3-642-19913-4

User-Centric Technologies and Applications, 2011
ISBN 978-3-642-19907-3

Vol. 95. Robert Burduk, Marek Kurzyński, Michał Woźniak, and Andrzej Żołnierek (Eds.)
Computer Recognition Systems 4, 2011
ISBN 978-3-642-20319-0

Vol. 96. A. Gaspar-Cunha, R. Takahashi, G. Schaefer, and L. Costa (Eds.)
Soft Computing in Industrial Applications, 2011
ISBN 978-3-642-20504-0

Dependable Computer Systems, 2011
ISBN 978-3-642-21392-2
Dependable Computer Systems
Preface

We would like to present the monographic studies on selected problems of dependable computer systems and networks which are published in the series *Advances in Intelligent and Soft Computing*.

The systems under consideration

Contemporary systems are created as very sophisticated products of human ideas and they are characterized by complex structure. The three main elements that should be identified in any system are: users, services (functionalities), and technological resources. The technological resources are understood as technical assets (engineering stuff) and information resources (algorithms, processes and management procedures). In the most general operation flow the users generate tasks which are realized by the system. The task to be realized requires some services (functionalities) available in the system and realization of the services needs a defined set of technical resources. In a case when any resource component of this set is in the state “out of order” or “busy”, the task may wait until the moment when the component returns to the state “available”, or the service may try to create other configuration based on available technical resources.

The modern systems are equipped with suitable measures which minimise the negative effects of these inefficiencies (a check-diagnostic infrastructure, fault recovery, information renewal, time and hardware redundancy, reconfiguration or graceful degradation, restart etc). The special service resources (service persons, different redundancy devices, etc.) supported by so called maintenance policies (procedures of resource services used to minimize negative consequences of faults that are prepared before or created ad hoc by the system manager) are incorporated in every real system.

System dependability

The dependability is a modern approach to reliability problems of contemporary systems. It is worth to underline the difference between the terms of system dependability and systems reliability. Dependability of systems, specially computer systems and networks, is based on a multi-disciplinary approach to theory, technology, and maintenance of systems operating in real (and very often unfriendly) environment. Dependability of systems concentrates on a probability of tasks realization by a system which is the unity of technical, information and human resources, while “classical” reliability focuses mainly on technical system resources.
The system dependability can be described by such attributes as availability (readiness for correct service), reliability (continuity of correct service), safety (absence of catastrophic consequences for the users and the environment), security (availability of the system only for authorized users), confidentiality (absence of unauthorized disclosure of information), integrity (absence of improper system state alterations) and maintainability (ability to undergo repairs and modifications).

System dependability and soft computing

The following main assumptions are usually made during dependability analyses and syntheses: system user tasks are realized on the basis of available services (functionalities) and information or technical resources. This means that the realized task is dynamical mapped on the services and then the services are dynamical mapped on the system resources. The system operates in unfriendly environment and its components (services and resources) are working with limited performance and with unreliable parameters, so consequently the user task is executed with limited performability parameters too. Sometimes a combination of unfriendly conditions of the environment together with possible faults produced by the technical infrastructure and/or by the users may create a critical situation in system operation which may lead not only to incorrect task realization but even to system collapse.

The functional and reliability parameters of system functionalities (services) and resources are often tightly inter-dependent – hence dependability analysis or synthesis of such contemporary systems need adequate formal and mathematical models and calculation (evaluation) methods suitable for systems and processes which are created by mix of stochastic and deterministic events generated by hardware resources, information resources (algorithms and procedures of operations and system management) and human-factors (managers, administrators and users). It is very often difficult to find a relation between system elements and system events (the relation between reasons and results) and it is even more difficult to define mathematical models with “analytical” relationships between such phenomena as, for example, a system user (administrator) mistake and the time extension of task execution in a distant node of the system. Of course, these problems are not only associated with human factors but the same difficulties are generated by complexity of technological and information system resources too. The problems may be solved only by using artificial intelligence and soft computing methods and, in this situation, contemporary systems, especially computer systems and networks, are the actual examples of undependable artificial intelligence systems which are not formally and mathematically described yet.

The topical scope of the monograph

In the following few points we are presenting the main subject areas of the chapters selected for our monograph.

Methodology

Compliant methodology and tools to develop and manage development environments of IT security-enhanced products and systems for the purposes of their certification are standardized in the ISO/IEC 15408 Common Criteria. In chapter 1,
Białas presents results of research on how to develop the set of patterns for different kinds of evidences that should be delivered together with the IT product or system for independent evaluation of its dependability, safety, security etc. levels. The work summarizes the methodology with respect to the achieved and planned project results.

An original methodology for dependability analysis which uses semi-Markov models is described in chapter 16. In this approach the models represent equipment ageing and also incorporate various maintenance activities. Having available some basic representation it is possible to adjust its parameters so that it corresponds to some hypothetical new maintenance policy and then to examine impact that this new policy has on various reliability characteristics of the system. The work deals with a methodology of model adjustment and specifically investigates its one particular problem: avoiding probability saturation in cases when the tuning is aimed at reaching increased repair frequencies.

Mathematical models

A model of client – server computing system is proposed by Zuberek in chapter 23. The main problem is how to model services requested by clients. The author presents a component language for description of system elements (service, client) and proposes interleaving requests from different components to increment performance of the system.

Design models can be analyzed to predict whether the future system satisfies requirements prior to its implementation. In Ghezzi and Sharifloo’s work (chapter 4) it is proposed to use a model-driven approach in analyzing design models against non-functional requirements (reliability, performance, cost, and energy consumption). The authors also show how probabilistic model checking techniques can be used to achieve this purpose. In the method it is assumed that initially the software engineer describes the desired system functionalities using behavioural models that represent high-level scenarios specified by UML Sequence Diagrams.

Functional approach to dependable operation of the system considered as network of services is presented by Walkowiak and Michalska in chapter 21 and by Toporkov et al. in chapter 19. The latter work proposes a slot selection algorithm with non-rectangular time-slot window. The slot algorithms are used in economic models for independent job batch scheduling in distributed computing with non-dedicated resources. Economic models of scheduling are based on the concept of fair resource distribution between users and owners of computational nodes. They are effectively used in such spheres of distributed computing as Grid, cloud computing, and multiagent systems. Resource brokers usually implement some economic policy in accordance with the application-level scheduling concept. When establishing virtual organizations, the optimization is performed for the job-flow scheduling. Application of the set of specific rules leads to overall increase in the quality of service (QoS) and in efficiency of resource usage.

Duration graphs are an extension of timed automata and are suitable for modelling the accumulated times spent by computations in duration systems. Majdoub in chapter 12 proposes a framework for automatic generation of test cases directly from the specification model represented by states and trees.
Software

Nowadays computer systems fail mainly due to software faults. One of the main reasons of software failures is software ageing. The ageing is the progressive software performance degradation due to accumulation of error conditions that leads to system resource exhaustion. To counteract aging, software rejuvenation has been recently proposed. Rejuvenation is the concept of periodically stopping the software, cleaning its internal state and then restarting. Koutras in chapter 8 examines how software availability is affected by rejuvenation technology and tries to find an optimal rejuvenation policy in terms of availability and downtime cost.

Effective testing is one of the key issues in development of dependable software systems. The objective of Bluemke and Kulesza’s work in chapter 2 is to compare dataflow and mutation testing of several Java programs. Results of experiments conducted in the Eclipse environment are also included.

Computer oriented languages are very often used in system modelling and the examples can be found in chapters 2, 4, 9, 10, 18 and 13. The UML formalization is often undertaken by projecting the notation in a rigorously defined semantic domain. When the target formalism is of state transition type, the derived models are verified by model checking. Meziani and Bouabana-Tebibel verify in chapter 13 a model using the Petri nets approach. The work of Kowalski and Magott in chapters 9 and 10 proposes UML models of maintenance policies which are built on the base of fault trees and Petri nets. The problem of developing dependable software (a metamodel and an UML profile for functional programming languages) is discussed by Szlenk in chapter 18.

Tools and technologies for dependable system operation

In many applications microcontroller circuits are used for improving dependability of reactive systems with real time requirements. In order to evaluate their fault susceptibility various fault injection techniques have been developed. The software for fault injection analysis applied for a case of satellite microcontroller on-board power system is presented by Iwiński and Sosnowski in chapter 5.

Memory failures are quite common in contemporary technology. When they occur, the whole memory bank has to be replaced, even if only few bytes of memory are faulty. With increasing sizes of memory chips the urge not to waste these “not quite properly working” pieces of equipment becomes larger and larger. Operating systems such as Linux already provide mechanisms for memory management which could be utilized to avoid allocating bad memory blocks which have been identified earlier, allowing for a failure-free software operation despite hardware problems. Surmacz and Zawistowski describe in chapter 17 problems of detecting memory failures and deal with OS (Linux) mechanisms that can be used for bad block exclusion.

The crucial role of evaluation during development of the information retrieval systems is to provide useful evidence of performance improvements and the quality of results that they return. However, the classic evaluation approaches have limitations and shortcomings especially regarding the user consideration, the measure of the adequacy between the query and the returned documents and the consideration of characteristics, specifications and behaviours of the search tool.
Bouramoul et al.’s chapter 3 presents a new approach for Web search engines evaluation that takes into account the context during the assessment process. The experiments included at the end prove the applicability of the proposed approach to the real research tools.

Theory of the systems reliability is particularly applicable to electronic protection systems (alarm systems) which, due to their specific character of use, should be characterized by the high level of reliability. The devices and electronic units applied in the wide range in those systems, in particular the microprocessor systems, require a new perspective on the system reliability and safety. The Sergiejczyk and Rosinski’s chapter (15) presents a reliability analysis of the electronic protection systems using optical links. Some theoretical and practical problems of detecting encrypted files in evidence data during digital forensics investigations are presented by Jóźwiak et al. in chapter 6.

**Probabilistic assessment methods**

One of the most difficult problem in development of the critical computer based systems is evaluation of system measures that are strongly related to probabilistic description of the events in the considered system. Sharvia and Papadopoulos (chapter 14) propose an approach to safety analysis which systematically integrates both compositional and behavioural safety analysis paradigms. The process starts with compositional analysis and uses its results to provide a systematic construction and refinement of state machines, which can be subsequently analyzed by behavioural analysis. Kharchenko et al. in chapter 7 try to built a technique for assessment of probabilistic dependability metrics of systems with different structures and failures. The technique is useful for synthesis of systems with applied redundancy and FTC concepts.

**Summary**

The brief overview of the chapters above illustrates a wide diversity of dependability problems in contemporary technical systems. We believe that this monograph will be interesting to all scientists, students, practitioners and researchers who deal with problems of dependability on practical grounds. It is our hope that it may be an inspiring source of original ideas and can help to define new challenges, as well as can provide a general insight into selected topics of the subject. As the editors we would like to express our sincere gratitude to all authors who have contributed to this volume and to all reviewers whose remarks has helped us to refine its contents.

Wojciech Zamojski
Janusz Kacprzyk
Jacek Mazurkiewicz
Jarosław Sugier
Tomasz Walkowiak
Patterns Improving the Common Criteria Compliant IT Security Development Process ................................................. 1 Andrzej Biłas

A Comparison of Dataflow and Mutation Testing of Java Methods .................................................................................. 17 Ilona Bluemke, Karol Kulesza

A New Three Levels Context Based Approach for Web Search Engines Evaluation .................................................. 31 Abdelkrim Bouramoul, Mohamed-Khireddine Kholladi, Bich-Lien Doan

Quantitative Verification of Non-functional Requirements with Uncertainty ................................................................. 47 Carlo Ghezzi, Amir Molzam Sharifloo

Testing Fault Susceptibility of a Satellite Power Controller ....................................................................................... 63 Marcin Iwiński, Janusz Sosnowski

Theoretical and Practical Aspects of Encrypted Containers Detection - Digital Forensics Approach .............................. 75 Ireneusz Jozwiak, Michal Kedziora, Aleksandra Melinska


Two-Level Software Rejuvenation Model with Increasing Failure Rate Degradation ................................................. 101 Vasilis P. Koutras
<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Towards a UML Profile for Maintenance Process and Reliability Analysis</td>
<td>117</td>
</tr>
<tr>
<td>Marcin Kowalski, Jan Magott</td>
<td></td>
</tr>
<tr>
<td>Conjoining Fault Trees with Petri Nets to Model Repair Policies</td>
<td>131</td>
</tr>
<tr>
<td>Marcin Kowalski, Jan Magott</td>
<td></td>
</tr>
<tr>
<td>Analysis of Geometric Features of Handwriting to Discover a Forgery</td>
<td>145</td>
</tr>
<tr>
<td>Henryk Maciejewski, Roman Ptak</td>
<td></td>
</tr>
<tr>
<td>A Formal Framework for Testing Duration Systems</td>
<td>155</td>
</tr>
<tr>
<td>Lotfi Majdoub</td>
<td></td>
</tr>
<tr>
<td>Dynamic Model Initialization Using UML</td>
<td>169</td>
</tr>
<tr>
<td>Lila Meziani, Thouraya Bouabana-Tebibel</td>
<td></td>
</tr>
<tr>
<td>Integrated Application of Compositional and Behavioural Safety Analysis</td>
<td>179</td>
</tr>
<tr>
<td>Septavera Sharvia, Yiannis Papadopoulos</td>
<td></td>
</tr>
<tr>
<td>Reliability Analysis of Electronic Protection Systems Using Optical Links</td>
<td>193</td>
</tr>
<tr>
<td>Mirosław Siergiejczyk, Adam Rosiński</td>
<td></td>
</tr>
<tr>
<td>Avoiding Probability Saturation during Adjustment of Markov Models of Ageing Equipment</td>
<td>205</td>
</tr>
<tr>
<td>Jarosław Sugier</td>
<td></td>
</tr>
<tr>
<td>Bad Memory Blocks Exclusion in Linux Operating System</td>
<td>219</td>
</tr>
<tr>
<td>Tomasz Surmacz, Bartosz Zawistowski</td>
<td></td>
</tr>
<tr>
<td>Metamodel and UML Profile for Functional Programming Languages</td>
<td>233</td>
</tr>
<tr>
<td>Marcin Szlenk</td>
<td></td>
</tr>
<tr>
<td>Resource Co-allocation Algorithms for Job Batch Scheduling in Dependable Distributed Computing</td>
<td>243</td>
</tr>
<tr>
<td>Victor Toporkov, Dmitry Yemelyanov, Anna Toporkova, Alexander Bobchenkov</td>
<td></td>
</tr>
<tr>
<td>Functional Based Reliability Analysis of Web Based Information Systems</td>
<td>257</td>
</tr>
<tr>
<td>Tomasz Walkowiak, Katarzyna Michalska</td>
<td></td>
</tr>
</tbody>
</table>
Human Resource Influence on Dependability of Discrete Transportation Systems ........................................ 271
Tomasz Walkowiak, Jacek Mazurkiewicz

An Effective Learning Environment ........................................ 285
Marek Woda, Konrad Kubacki-Gorwecki

Incremental Composition of Software Components .......... 301
W.M. Zuberek

Author Index ................................................................. 313
Patterns Improving the Common Criteria Compliant IT Security Development Process

Andrzej Białas

Institute of Innovative Technologies EMAG, 40-189 Katowice, Leopolda 31, Poland
e-mail: a.bialas@emag.pl

Abstract. The chapter concerns the project of the methodology used to create and manage development environments of IT security-enhanced products and systems for the purposes of their future Common Criteria certification. The key issues of the patterns-based project are discussed: how to develop the set of patterns for different kinds of evidences to be delivered with the IT product or system for independent evaluation. The author characterizes the IT security development process and the elaborated evidences, and presents analyses provided to develop such patterns. The patterns usage is shown by a few examples which are part of a more complex case study. Such patterns facilitate and speed up the IT security development process, improve the quality of evaluation evidences, as they are more consistent and include all details required by the considered assurance components, facilitate the computer support of the IT security development process. The chapter concludes the methodology with respect to the achieved and planned project results.

1 Introduction

Information technologies (IT/ICT) are often meant to fulfil social and business objectives in high-risk environments. The main issue, related to the term “assurance”, is which factors ensure that the measures meant to protect certain resources in a critical situation will really work. The assurance is understood that an IT product or system meets its security objectives expressing these measures. The basic assurance methodology is specified within the ISO/IEC 15408 Common Criteria (CC) standard [14]. According to the CC paradigm the source of measurable assurance are the rigour applied during the development and manufacturing processes, independent third-party evaluation, operation and maintenance according to the received certificate. The reliable IT products and systems with measurable level of assurance are developed in a rigorous manner in special “development environments” (exactly: development, production and maintenance environments). Assurance is measured with the use of Evaluation Assurance Levels (EALs) in the range: EAL1 (min.) to EAL7 (max.).

The chapter presents the concept, first results and future plans of the CCMODE (Common Criteria compliant, Modular, Open IT security Development Environment)
R&D Project carried out by the Institute of Innovative Technologies EMAG [13]. CCMODE is co-financed by the EU within the European Fund of Regional Development. The objective of the project is to work out a CC-compliant methodology and tools to develop and manage development environments of IT security-enhanced products and systems for the purposes of their future certification. The basic products of the project will be the following: knowledge, patterns (including documentation, procedures, evidences, etc.), methodology and tools used to create and manage development environments by different business organizations. The IT products and systems developed in these environments, having measurable assurance (EAL), can be certified and used in high risk applications.

CCMODE assumes participation in activities dealing with the Common Criteria methodology improvement and extending the range of the standard application:

1. The project does not provide a solution dedicated to one environment only. It provides patterns and methods for developing a wide range of environments – generally where the Common Criteria standard is applicable. For this purpose a set of modules will be developed, along with the methodology of adapting them to the needs of a given environment. The possible reusing of the same patterns many times will bring significant financial benefits.

2. The project will provide an integrated solution (based on ISO/IEC 15408 and ISO/IEC 27001) for the IT security management system, dedicated to development environments. The solution will enable better protection of information related to projects carried out in these environments and will consider business continuity aspects of manufacturing and maintenance processes.

3. The project considers both – traditional “sites” organized to develop a certain IT product or system, and sites compliant with the “site-certification” concept [24], allowing to reuse some evidences in the elaboration of a certain group of similar IT products or systems.

4. The project will provide a computer tool which will support even the most laborious and difficult operations related to the management of the environment and product or system in the life cycle (security analyses, management of evidences, configuration, flaws and tools). This will bring extra benefits to organizations in the form of management processes automation.

5. The project is strongly based on the life cycle model, knowledge engineering and risk analysis methods.

The project has its national aspect – disseminating and finally implementing the ISO/IEC 15408 Common Criteria (CC) standard in Poland, creating the CC community and elaborating best practices.

The chapter gives a short introduction to the project domain, surveys current state of the works, discusses the key issues of the project: how to develop the set of patterns for the security specification means and for the evaluation evidences to be delivered with the IT product or system for independent evaluation. All kinds of patterns are identified and presented. Some of them are shown by examples. The last section concludes the project efforts and specifies the planned works.
2 Project Background

The project background is founded on the Common Criteria methodology specified within the [14]. The project concerns the development-, production- and maintenance processes of any IT product or system which should be protected against threats. In practice the Common Criteria standard, as well as the CCMODE project, concerns all kinds of IT products or systems, called TOEs (Targets of Evaluation). The TOE can be: hardware, software, combination of the above, application programs, tools for products development, complete systems, etc. More than 1200 TOEs have been developed and evaluated so far [15].

2.1 The Common Criteria IT Security Development Methodology

The Common Criteria methodology [14] encompasses the three main processes:

- the IT security development process, related to the elaboration of the document called ST (Security Target) or PP (Protection Profile) for the given TOE, presenting the TOE, specifying its security problem definition, security objectives, requirements, and finally the TOE security functions for ST which are later implemented on the claimed EAL level;
- the TOE development process, related to the IT product or system development with the use of the assumed technology, including its TOE security functions implementation on the claimed EAL level;
- the IT security evaluation process, performed by an independent, accredited security lab – the process completed by the certification; during this process the TOE, its Security Target and evidences are evaluated according to the CC methodology with respect to the declared EAL requirements.

The Common Criteria (CC) methodology is matured but still improved. The main challenges are: raising the design preciseness, facilitating the development and evaluation, decreasing the development cost and time. The general motivation of the author’s works is to improve the IT security development process using the advantages and new possibilities offered by the patterns-based approach, and to minimize the barrier for developers related to the lack of knowledge, methods and exemplars of evidences, etc.

Thanks to the patterns, the evidences can be elaborated in a unified proven way and developers can acquire knowledge how to elaborate evidences. More information about the Common Criteria methodology can be found in [15], [18], [9], [2].

2.2 Patterns-Based Development

Design patterns are often used by engineers in many technology domains, including here discussed IT- and IT security domains. These patterns can be considered as reusable, proven solutions to problems with respect to a specific context. They
provide development process knowledge helping to achieve the expected solution within the project domain. Patterns-based development enforces standard solutions, saves time and money and improves quality.

The patterns are specified in a formalized way, e.g. using UML (Unified Modelling Language), OCL (Object Constraints Language), different kinds of codes, ontologies, formalized descriptors, etc. Numerous patterns concern software architecture, including applied security mechanisms.

The paper [25] surveys approaches to the security patterns categorized with respect to the software life cycle phases, i.e. requirements-, design- and implementation phases. No approaches closely related to the Common Criteria methodology were encountered in this paper.

The commonly used security design patterns related to software solutions are specified in [23]. These architectural patterns related to the enterprise management applications concern, for example: enterprise security and risk management, identification and authentication, access control models and systems, operating system access control, accounting facilities, firewall architecture, secure Internet applications, IP telephony, and cryptographic key management. With respect to the CC methodology, this kind of security design patterns can be used to implement the security functions (on a given EAL) within an IT product or system.

The book [19] introduces a UML extension called UMLsec. It provides a unified approach to security features description during the secure systems development. UMLsec allows to define the UML patterns that encapsulate the design knowledge in the form of recurring design problems, and consist of the “pattern name”, “problem description”, “problem solution”, and “consequences”. The established formal rules of security engineering can be used by a wider group of developers. The elaborated patterns present the basic security engineering solutions, like: electronic signature, secure Java programs, electronic purse, secure channel, TLS Internet protocol, bank applications, biometric authentication systems, etc. Please note that these patterns focus on modelling IT security features and behaviour within the system. The UMLsec patterns defined in [19] do not concern the IT security development process compliant with the Common Criteria standard, although they can be helpful in this process, e.g. to evaluate UML specifications against different vulnerabilities. This issue was discussed in the monograph [9].

The author’s preliminary researches discussed in the Section 2.3 concern security specification means patterns. They are not focused on IT security functionality, as the above mentioned security design patterns, but rather on risk management- and assurance issues closely related to the Common Criteria methodology. From this point of view they differ from the above mentioned on one hand, and supplement them on the other hand. The security specification means patterns are related to the IT security development process mentioned in the Section 2.1.

CCMODE adopts the specification means patterns, though it is focused on the development of a new class of the CC-related patterns – patterns of the evaluation evidences, related mainly to the TOE development process. There are no researches
focused on the elaboration of evaluation evidences patterns. Some guidance is provided by the standard [14] and BSI guide [17] restricted to EAL 1-5.

### 2.3 The CCMODE Project Background and Current Results

The project background was based on the preliminary, multidirectional researches. The first direction was focused on the UML/OCL modelling. The Common Criteria compliant, UML/OCL-based IT security development framework (ITSDF) [9], [3], [10], [8] was elaborated, which embraces:

- models of data structures and processes of IT security development stages, including: security problem definition (SPD), security objectives (SOs) elaboration, security functional requirements (SFRs), and TOE security functions (TSFs) workout; two kinds of classes are distinguished: classes responsible for the ST elaboration and classes representing specification data containers;
- models of the specification means used for these IT security development stages, including CC security components – functional- (SFRs) and assurance components (SARs) [14] and the semiformal enhanced generics; please note that generics and components constitute a security specification language.

The semiformal ITSDF framework was implemented as a software tool to support IT security developers. Please note that the introduced enhanced generics are derived from “generics” commonly used by IT security developers. The enhanced generics are defined as mnemonic names expressing common features, behaviours or actions related to IT security issues, like: subjects (active entities), objects (passive entities), threats, assumptions, organizational security policies (OSP), security objectives, and security functions [3], [9]. They are “enhanced” since they are semiformal and have features comparable to CC components, allowing such operations as: parameterization, derivation, iteration, and refinement. Enhanced generics can be grouped by domains of applications. They allow to create generics chains which express solutions to elementary security problems. This way the preliminary version of the Security Target (ST) pattern and specification means patterns used to fulfil this ST structure was elaborated.

The second direction of the preliminary researches deals with the application of the ontological approach [21], [22] to this framework [4], [7]. Please note that the ontology represents explicit formal specifications of the terms in the project domain and relations between these terms. The elaborated ITSDO (IT Security Development Ontology) represents the security requirements structures (i.e. ST, PP), specification means to fill in these structures with contents for different TOEs (author’s defined enhanced generics, CC-defined functional and assurance components) as well as patterns for evidences. During these works about 350 enhanced generics were predefined as elementary items designed to specify general security features of commonly used IT products or systems.

The third direction of preliminary researches is focused on the validation of the specification means patterns on different TOE designs (firewall [12], motion
sensor [5], [6], medical sensor [2], gas detecting sensor [1]), their improvement and the elaboration of patterns for selected evidences.

CCMODE should bring these partial results together, supplement them and create a complex and unified CC related patterns system facilitating the IT security development. To achieve this, the following 7 project tasks were scheduled:

1. Research of development environments of IT security-enhanced products with respect to life cycle processes – identification of developers’ needs and expectations, building a reference model.
2. Working out an open, modular development (manufacturing, maintenance) environment – patterns developed for evidence material related to the environment and to the product developed in this environment.
3. Working out a methodology for implementation and management of development environments – key issues: how to adapt open patterns to the identified needs, how to develop an environment based on these needs, and how to manage this environment.
4. Validation of the implementation methodology – with the participation of security developers and evaluators.
5. Working out a tool to support the development environment management – automation of difficult and repeatable operations.
6. Validation of the tool supporting the management of development environments – with the participation of security developers and evaluators.
7. Creating an experimental development environment – also: promotion, CMODE community.

The task 1 focused on researches of development environments of IT security-enhanced products or systems with respect to the life cycle processes. The knowledge was acquired about the structure, operation of development environments of security-enhanced products. The assurance components were analyzed in terms of their relationships with the development environment and the IT product or system developed in the environment. On this basis the reference model of the development environment with an embedded life cycle was made. Then the method to assess the compliance of any development environment with the CC-based reference model was elaborated and validated with the participation of IT developers.

The task 2, closely related to this chapter, was focused on working out an open, modular development (manufacturing, maintenance) environment, represented by a set of Common Criteria related patterns, i.e.:

- SST (Site Security Target) document pattern for a local development environment according to the site certification approach,
- evidences patterns to evaluate the environment based on the requirements of the ALC (Life cycle support) assurance class,
- pattern modules of the information security management system compliant with ISO/IEC 27001 for the development environment,
- security specification patterns in the form of Security Target (ST) and Protection Profile (PP), including their low-assurance versions,
- evidences patterns for IT products and systems developed in the environment.
Currently the set of patterns of the evaluation evidences as well as the specification means are elaborated and the third task of the project has been carried out – focused on the implementation of the development environments. The gap analysis method has been elaborated. The organization which plans this implementation should be first audited to identify its needs, restrictions and any incompliance with the Common Criteria standard. The gap analysis gives input to the adaptation of patterns for the created development environment. The customized patterns define the development-, production- and maintenance processes of the organization, reflecting the assumed life cycle model and helping to manage this environment.

3 Patterns in the CCMODE Project

There are three kinds of patterns considered in the project: patterns of evaluation evidences, patterns of specification means, and patterns of auxiliary documents.

The patterns of evaluation evidences required by the standard and delivered for the independent evaluation are of key importance for the CCMODE project. They play the roles of “open modules” that should be refined and customized according to the developers’ needs and expectations to implement the development environment for the given IT product or system, and to elaborate evaluation evidences for the given TOE. They will be discussed in the next subsection.

The patterns of the specification means include the author’s defined enhanced generics and standard-defined functional and assurance components. They can be expressed informally using the dot separated notation, semi-formally using the UML notation or as knowledge base items. Currently the enhanced generics are validated on different designs and optimized. Moreover the generics subsets for particular domains of application are elaborated, e.g. for intelligent sensors [1], [2]. The patterns of the specification means are omitted in this chapter because the author’s many publications have discussed this issue.

During the development-, manufacturing- and maintaining processes as well as the evaluation process many auxiliary documents, procedures, forms, checklists, reports, etc. can be useful. Many patterns belonging to this family are related to the ISMS (Information Security Management System) implementation within the development environment. These patterns will be omitted in this chapter.

3.1 Identification of the Evaluation Evidences Patterns

Two important issues were solved during the initial tasks of CCMODE: how many patterns of the evaluation evidences are needed and what their organization and shape look like.

Analyzing the contents of the assurance classes components, five groups of evaluation evidences patterns were distinguished:

- patterns for basic security requirements,
- patterns of evaluation evidences closely related to the development environment, its organization and development-, production- and maintenance processes,
patterns of evaluation evidences closely related to the IT product or system (TOE),

patterns of evaluation evidences related to both the development environment and the IT product or system,

patterns of evaluation evidences related to the composition.

Basic security requirements are considered as evaluation evidences as well. The following five patterns were defined:

- Security Target pattern (STp), low assurance Security Target pattern (laSTp) used for EAL1 – both based on the ASE (Security Target Evaluation) assurance class,
- Protection Profile pattern (PPp), low assurance Protection Profile pattern (laPPp) used for EAL1 – both based on the APE (Protection Profile Evaluation) class,
- Site Security Target pattern (SSTp) – based on the AST (Site Security Target Evaluation) class defined outside the [14], i.e. in the [24] guide.

The first four of them contain basic security requirements for the TOE, while the latter one, concerning the site certification, deals with the site, considered as the development environment. Each Common Criteria project uses one of the basic security requirements. This main document helps to organize other evidences. Site Security Target is optional.

The second group of patterns is closely related to the requirements for the development environment, its organization and its development-, production- and maintenance processes. All these issues are specified in seven assurance families belonging to the ALC (Life-cycle support) assurance class:

- Life-cycle model definition pattern (ALC_LCDp) presents the high-level description of the TOE life-cycle and creates the framework for the entire development environment;
- Development security pattern (ALC_DVSp) presents physical, procedural, personnel, and other measures used in the development environment to protect the TOE and its parts; it can be extended by the Information Security Management System pattern (ISMSp) compliant with ISO/IEC 27001; ISMSp represents an extensive set of patterns needed to implement the standard in the development environment to better protect design data and manage information security;
- Configuration management (CM) capabilities pattern (ALC_CMCp) defines a more detailed description of the management of the configuration items and enforces discipline and control in the processes of refinement and modification of the TOE and the related information;
- Configuration management scope pattern (ALC_CMSp) shows how to specify items to be included as configuration items and hence controlled by the above CM capabilities (ALC_CMCp);
- Tools and techniques pattern (ALC_TATp) is responsible for control tools, their options and techniques used in the development environment (programming languages, documentation, implementation standards, runtime libraries, different equipment, etc.);