OPBUS: A Framework for Improving the Dependability of Risk-Aware Business Processes

Ángel Jesús Varela Vaca, 28.808.965-Q

ajvarela@us.es

Supervised by Prof. Dr. Rafael Martínez Gasca

Thesis Dissertation submitted to the Department of Computer Languages and Systems of the University of Seville in partial fulfilment of the requirements for the degree of Ph.D. in Computer Science.

(Thesis Dissertation)
Abstract

There currently exists a growing trend to externalize and automate increasingly complex and critical processes of organization services and operations by using business process management systems. These processes are ever threatened by security risks due to the exposure of business processes to external and third parties outside the control of the organizations. Thereby, there exists a growing interest in improving the quality of these processes in order to ensure conformance and compliance in the management of security risks during all the phases of business process life-cycle. Nevertheless, there is a need for comprehensive framework for automatically managing and ensuring compliance and conformance of security risk requirements. To this end, this thesis dissertation propose a framework (OPtimization of BUsiness process Security, OPBUS) for the improvement of the dependability of the life-cycle of business process management in both design and run-time stages. The framework presents an enhancement of the business process management life-cycle with risk management capabilities at design time and fault tolerance at run-time. The main contributions presented in this dissertation include: (1) diagnosis of non-conformance of risks in business process models; (2) selection and generation of optimal security countermeasures for business processes in order to deal with previously detected non-conformance; (3) provision of a fault-tolerance infrastructure in order to ensure the correct execution in spite of the existence of faults.

More precisely, in the first place, we propose a fully automated approach focused on verification methods for the risk assessment of business process models, and the diagnosis of non-conformance of risks with regard to an acceptable risk threshold established in objectives. To this end, we propose an extension to business process models that enables the risk identification and risk estimation of business process models. Secondly, the automated diagnosis is carried out using Constraint Programming based on Artificial Intelligence techniques. The proposal has been supported with the development of tools that enable the graphical specification of the extension of business process, the automatic transformation of the model to constraint satisfaction problems and the automation of the diagnosis process through constraint solvers.

Secondly, a risk treatment must be defined once the non-conformance of risks are identified. Therefore, one of the main aims of this dissertation is to provide an automated approach to select and generate countermeasures in order to fix the non-conformances. We provide a formalization of IT security countermeasures based on security patterns and feature models for the representation countermeasures. We have defined a catalogue of IT security countermeasures to enforce confidentiality, integrity, availability and authentication in business processes. Furthermore, we have proposed automatic techniques...
based on Feature-Oriented Domain Analysis (FODA) and Constraint Programming for the inference, selection, and generation of optimal configurations with regard to single and multiple objectives.

And finally, despite countermeasures, business processes are no fault-free (such as zero-day vulnerabilities). It is crucial to provide business processes with fault-tolerance at run-time that ensure the correct execution of business process in spite of faults. This thesis dissertation provides a fully automated fault tolerance approach for business process in run-time. This approach consists of various fault tolerant patterns for the implementation of executable business processes. These patterns have been equipped with various recovery mechanisms based on: replication (dynamic binding of services); roll-back and check-pointing, and software diversity (NV-Versioning) techniques. The contribution also provides error detection mechanisms. The error detection mechanisms have been enhanced with a diagnoser which uses model-based diagnosis techniques in order to identify and isolate faulty services. Diagnosis has been developed using constraint programming techniques that enables the automatic identification of faulty services at run-time. Subsequently, the faulty services are reported to recovery mechanisms that are responsible for triggering an alternative service by means of the corresponding fault tolerance patterns.
Hoy en día existe una creciente tendencia a externalizar y automatizar procesos críticos y complejos mediante el uso de sistemas de gestión de procesos de negocio. Estos procesos son cada vez más frecuentemente amenazados por riesgos de seguridad debido a su exposición por entidades externas y servicios de terceros fuera del control de las organizaciones. De este modo, existe un creciente interés en la mejora de la calidad de estos procesos de negocio con el fin de garantizar el cumplimiento y la gestión de la conformidad de los riesgos de seguridad durante todas las fases del ciclo de vida de los procesos de negocio. Sin embargo, hay una clara ausencia de un marco común para la gestión automática y el aseguramiento del cumplimiento y la conformidad de los requisitos relacionados con los riesgos de seguridad. Para este fin, esta Tesis Doctoral propone un framework (OPtimization of BUsiness process Security, OPBUS) para mejorar la confiabilidad del ciclo de vida de gestión de procesos de negocio, tanto en fases de diseño como en tiempo de ejecución. El framework presenta una mejora de la gestión de procesos de negocio del ciclo de vida por medio de la gestión de riesgos en tiempo de diseño y tolerancia a fallos en tiempo de ejecución. Las principales contribuciones presentadas en esta tesis son: (1) diagnóstico de la no conformidad de los riesgos en los modelos de procesos de negocio, (2) la selección y generación de contramedidas de seguridad óptimas para el proceso de negocio con el fin de tratar las no conformidades detectadas; (3) aprovisionamiento de una infraestructura de tolerancia a fallos en tiempo de ejecución con el fin de asegurar el correcto funcionamiento de los procesos a pesar de los fallos.

Más precisamente, en un primer lugar, se propone una aproximación totalmente automatizada centrado en métodos de verificación para la evaluación de riesgos de los modelos de procesos de negocio, y el diagnóstico de las no conformidades de los riesgos con respecto a un umbral de riesgo acceptable establecidas en los objetivos. Para este propósito, primero hemos definido una extensión de los modelos de procesos de negocio que permite la identificación y estimación de los riesgos. En segundo lugar, el diagnóstico automatizado se lleva a cabo utilizando técnicas basada en Inteligencia Artificial como son las técnicas de programación con restricciones. La propuesta ha sido apoyada con el desarrollo de herramientas que permiten la especificación gráfica de la extensión del proceso de negocio, la transformación automática del modelo a problemas de satisfacción de restricciones y la automatización del proceso de diagnosis a través de diferentes resolutores.

En segundo lugar, una vez que las no conformidades de los riesgos se identifican debe definirse un tratamiento de estos riesgos. Por lo tanto, uno de los principales objetivos de esta tesis es proporcionar un método automático para seleccionar y generar contramedidas con el fin de corregir las no conformidades. En primer lugar proporcionamos una formal-
ización para contramedidas en base a patrones de seguridad y modelos de características. Hemos definido un catálogo de contramedidas de seguridad para hacer cumplir con la confidencialidad, integridad, disponibilidad y autenticación en los procesos de negocio. Además, se han propuesto técnicas automáticas basadas en Feature-Oriented Analysis Domain (FODA) y programación con restricciones para la inferencia en la selección y la generación de configuraciones óptimas con respecto a uno o múltiples objetivos.

Por último, a pesar de las contramedidas, los procesos de negocio no están libre de fallos (por ejemplo, las vulnerabilidades zero-day ). Es fundamental proporcionar a los procesos de negocio con tolerancia a fallos que garanticen la correcta ejecución de procesos de negocio en tiempo de ejecución, a pesar de las fallos. Esta tesis doctoral proporciona una aproximación totalmente automatizado de tolerancia a fallos para procesos de negocio en tiempo de ejecución. Esta aproximación consiste en varios patrones de tolerancia a fallos para la procesos de negocio. Estos patrones han sido equipados con diferentes mecanismos de recuperación basados en: replicación (enlace dinámico de los servicios), roll-back y el check-pointing, y técnicas de diversidad de software (NV-versioning). La contribución también proporciona mecanismos para la detección de errores. Los mecanismos de detección de errores se han mejorado con un diagnosticador mediante técnicas de diagnosis basado en modelos (MDB) con el fin de identificar y aislar los servicios defectuosos. MDB se han desarrollado utilizando programación con restricciones para la identificación automática de los servicios defectuosos en tiempo de ejecución. Posteriormente, los servicios defectuosos son reportados a los mecanismos de recuperación que son responsables desplegar una alternativa para estos servicios por medio de los patrones de tolerancia a fallos correspondientes.
## Contents

### List of Figures xi

### List of Tables xiii

### I  Preface 1

1. **Introduction** ........................................ 3
   - 1.1  Context and Motivation ................................ 5
   - 1.2  Problem Statements .................................. 7
   - 1.3  Thesis Context and Published Results .................. 9
   - 1.4  Roadmap: Structure of the Thesis ....................... 12

### II  Background 15

2. **Foundations** ......................................... 17
   - 2.1  Business Process Concepts ................................ 17
     - 2.1.1  Business Process Modelling .......................... 18
     - 2.1.2  Business Process Analysis ........................... 20
   - 2.2  Risk Management ...................................... 21
     - 2.2.1  Risk Assessment .................................... 23
     - 2.2.2  Risk Treatment ..................................... 23
     - 2.2.3  Risk Monitoring .................................... 24
   - 2.3  Model-based Fault Diagnosis ............................ 24
     - 2.3.1  DX: Logical Diagnosis Approach ..................... 25
     - 2.3.2  FDI: Analytical Redundancy Approach ................. 27
   - 2.4  Constraint Programming ................................. 30
     - 2.4.1  Searches ........................................... 31
     - 2.4.2  Optimization Problems ............................... 31
     - 2.4.3  Overconstrained Constraint Satisfaction Problems .... 32
   - 2.5  Feature-Oriented Domain Analysis ....................... 33
   - 2.6  Dependability and Fault Tolerance ...................... 34
<table>
<thead>
<tr>
<th>CONTENTS</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.4.3 Security pattern for Authorization</td>
<td>123</td>
</tr>
<tr>
<td>5.5 Automatic Selection and Generation of Security Configurations</td>
<td>127</td>
</tr>
<tr>
<td>5.6 Related Work</td>
<td>130</td>
</tr>
<tr>
<td>5.7 Summary</td>
<td>134</td>
</tr>
<tr>
<td>6 Dependability for Executable Business Processes</td>
<td>137</td>
</tr>
<tr>
<td>6.1 Context and Motivation</td>
<td>137</td>
</tr>
<tr>
<td>6.2 Fault Tolerance Layer for Executable Business Processes</td>
<td>138</td>
</tr>
<tr>
<td>6.2.1 Error detection component</td>
<td>140</td>
</tr>
<tr>
<td>6.2.1.1 Determine discrepancies</td>
<td>140</td>
</tr>
<tr>
<td>6.2.1.2 Model-based fault diagnosis</td>
<td>141</td>
</tr>
<tr>
<td>6.2.2 Recovery component</td>
<td>142</td>
</tr>
<tr>
<td>6.2.2.1 Replication and redundancy pattern</td>
<td>143</td>
</tr>
<tr>
<td>6.2.2.2 Diversity and N-Version Programming pattern</td>
<td>143</td>
</tr>
<tr>
<td>6.2.2.3 Check-pointing and Roll-Back pattern</td>
<td>145</td>
</tr>
<tr>
<td>6.3 Illustrative example</td>
<td>148</td>
</tr>
<tr>
<td>6.3.1 Performance Results</td>
<td>149</td>
</tr>
<tr>
<td>6.4 Related work</td>
<td>151</td>
</tr>
<tr>
<td>6.5 Summary</td>
<td>153</td>
</tr>
<tr>
<td>IV Conclusions and Future Work</td>
<td>157</td>
</tr>
<tr>
<td>7 Final Remarks</td>
<td>159</td>
</tr>
<tr>
<td>8 Directions of Future Work</td>
<td>163</td>
</tr>
<tr>
<td>8.1 Diagnosis of Non-Conformance of Risks in Business Process Models</td>
<td>163</td>
</tr>
<tr>
<td>8.2 Risk Treatment for Business Process Models</td>
<td>163</td>
</tr>
<tr>
<td>8.3 Dependability in execution of business</td>
<td>164</td>
</tr>
<tr>
<td>8.4 Risk Monitoring for Business Processes</td>
<td>164</td>
</tr>
<tr>
<td>V Appendices</td>
<td>167</td>
</tr>
<tr>
<td>A Transformation Template for Business Process Models</td>
<td>169</td>
</tr>
<tr>
<td>B Example of CSP</td>
<td>177</td>
</tr>
<tr>
<td>C Validation constraints for business process models</td>
<td>185</td>
</tr>
<tr>
<td>D Feature Analysis of SSL and TLS protocol</td>
<td>191</td>
</tr>
<tr>
<td>D.1 Constraint Model for Feature Model of SSL/TLS</td>
<td>193</td>
</tr>
<tr>
<td>E Feature Analysis of Web Application Firewall (WAF) Systems</td>
<td>203</td>
</tr>
<tr>
<td>F Feature Model for Fault Tolerance</td>
<td>209</td>
</tr>
</tbody>
</table>
List of Figures

1.1 Business process management landscape [Weske 2007] ........................................... 4
1.2 Lifecycle of Business Process Management. ............................................................ 6
1.3 Enhanced Business Process Management. ............................................................... 8
1.4 The Structure of the Thesis. ....................................................................................... 12
2.1 Example of business process model in BPMN notation. ........................................... 19
2.2 Overview of risk Management life-cycle. .................................................................. 22
2.3 Polybox system. .......................................................................................................... 26
2.4 Graphical solution of a CSP system ......................................................................... 30
2.5 Feature model example .............................................................................................. 34
2.6 Example of transformation of a feature model to a COP ........................................ 35
2.7 Dependability taxonomy [Pullum 2001]. .................................................................. 36
2.8 Basic design of diversity. ........................................................................................... 38
2.9 Basic RcB structure. ................................................................................................. 39
3.1 Extended Business Process Management. ................................................................. 47
3.2 Diagram with the main contributions of OPBUS. ...................................................... 48
4.1 Diagram with the process of diagnosis of non-conformance ........................................ 53
4.2 Business Process Meta-Model. ................................................................................. 55
4.3 Risk Model Extension. ............................................................................................... 57
4.4 Example of Business Process Model ........................................................................ 59
4.5 Estimation of elements in a business process model ................................................ 60
4.6 Threat Scenario attached to a business process model .............................................. 62
4.7 Evaluation threats without threat scenarios. ............................................................... 63
4.8 Threat scenario with countermeasures. ...................................................................... 66
4.9 Business process with objectives and acceptable risk .............................................. 68
4.10 Example of a business process model to be assessed. ............................................. 72
4.11 Business process with two activities in non-conformance to the acceptable risk level ............................................................................................................................................................................ 78
4.12 Code of the constraint model for the risk assessment of the integrity dimension .......... 79
4.13 Diagnosis process of checking conformance. .......................................................... 80
4.14 Model-Driven approach for transformation of process. ........................................... 81
4.15 OPBUS plugin architecture. ..................................................................................... 82
4.16 Integrated development of OPBUS environment. .................................................... 84
4.17 Properties dialogue. ................................................................................................. 85
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.18</td>
<td>Business process model of the example.</td>
<td>86</td>
</tr>
<tr>
<td>4.19</td>
<td>Potential execution flows of business processes.</td>
<td>90</td>
</tr>
<tr>
<td>4.20</td>
<td>Piece of generated COMET code.</td>
<td>91</td>
</tr>
<tr>
<td>4.21</td>
<td>Results of the diagnosis of non-conformance in the business process model.</td>
<td>92</td>
</tr>
<tr>
<td>5.1</td>
<td>Hierarchy of IT security controls for secure protocols.</td>
<td>100</td>
</tr>
<tr>
<td>5.2</td>
<td>Diagnosis of non-conformance of risk in a BPMS.</td>
<td>101</td>
</tr>
<tr>
<td>5.3</td>
<td>Application of countermeasures to $\Delta$.</td>
<td>104</td>
</tr>
<tr>
<td>5.4</td>
<td>Security Pattern meta-model.</td>
<td>108</td>
</tr>
<tr>
<td>5.5</td>
<td>Scenario of BPMS to be enforced by security countermeasures.</td>
<td>113</td>
</tr>
<tr>
<td>5.6</td>
<td>Feature model for fault tolerance.</td>
<td>117</td>
</tr>
<tr>
<td>5.7</td>
<td>Example of SSL/TLS configuration for a Jetty Server.</td>
<td>120</td>
</tr>
<tr>
<td>5.8</td>
<td>Feature model of SSL/TLS.</td>
<td>121</td>
</tr>
<tr>
<td>5.9</td>
<td>Example of rules for the detection of protocol anomalies in ModSecurity.</td>
<td>125</td>
</tr>
<tr>
<td>5.10</td>
<td>Feature model for WAF.</td>
<td>126</td>
</tr>
<tr>
<td>5.11</td>
<td>Process of selection of configurations.</td>
<td>128</td>
</tr>
<tr>
<td>6.1</td>
<td>Fault tolerance architecture perspective.</td>
<td>139</td>
</tr>
<tr>
<td>6.2</td>
<td>Process for error detection using model-based diagnosis.</td>
<td>140</td>
</tr>
<tr>
<td>6.3</td>
<td>Fault tolerance pattern with binder.</td>
<td>144</td>
</tr>
<tr>
<td>6.4</td>
<td>Fault tolerance pattern with replicated binder.</td>
<td>144</td>
</tr>
<tr>
<td>6.5</td>
<td>Fault tolerance pattern with N-Version components.</td>
<td>146</td>
</tr>
<tr>
<td>6.6</td>
<td>Fault tolerance pattern with check-points.</td>
<td>147</td>
</tr>
<tr>
<td>6.7</td>
<td>Scenario of example of distributed business processes.</td>
<td>148</td>
</tr>
<tr>
<td>6.8</td>
<td>I. Performance with one thread for execution. II. Performance with more than one thread for execution.</td>
<td>150</td>
</tr>
<tr>
<td>D.1</td>
<td>SSL handshake protocol actions [Stallings 2002].</td>
<td>192</td>
</tr>
</tbody>
</table>
## List of Tables

<table>
<thead>
<tr>
<th>Table Number</th>
<th>Table Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Signature matrix for single faults</td>
<td>29</td>
</tr>
<tr>
<td>4.1</td>
<td>Relations in the extension meta-model</td>
<td>58</td>
</tr>
<tr>
<td>4.2</td>
<td>Summary of the elements of extension for business processes.</td>
<td>69</td>
</tr>
<tr>
<td>4.3</td>
<td>Basic patterns for risk determination in business processes.</td>
<td>73</td>
</tr>
<tr>
<td>4.4</td>
<td>Activity abbreviations for the example.</td>
<td>86</td>
</tr>
<tr>
<td>4.5</td>
<td>Description of threats and countermeasures.</td>
<td>87</td>
</tr>
<tr>
<td>4.6</td>
<td>Acceptable risk threshold for business processes.</td>
<td>88</td>
</tr>
<tr>
<td>4.7</td>
<td>List of threats and values associated to the activities of the business process</td>
<td>88</td>
</tr>
<tr>
<td>4.8</td>
<td>Frequencies and consequences related to threats.</td>
<td>89</td>
</tr>
<tr>
<td>4.9</td>
<td>Risk Reduction values related to countermeasures.</td>
<td>89</td>
</tr>
<tr>
<td>4.10</td>
<td>Non-conformance solution for Hiring Manager.</td>
<td>89</td>
</tr>
<tr>
<td>4.11</td>
<td>Results of the diagnosis of potential execution flows.</td>
<td>90</td>
</tr>
<tr>
<td>4.12</td>
<td>Results of the non-conformance by activities.</td>
<td>93</td>
</tr>
<tr>
<td>4.13</td>
<td>Comparison of approaches</td>
<td>97</td>
</tr>
<tr>
<td>5.1</td>
<td>Summary of security pattern elements.</td>
<td>112</td>
</tr>
<tr>
<td>5.2</td>
<td>Security pattern for fault tolerance in business processes.</td>
<td>114</td>
</tr>
<tr>
<td>5.3</td>
<td>Description of features for fault tolerance.</td>
<td>115</td>
</tr>
<tr>
<td>5.4</td>
<td>Description of attributes for the feature model of fault tolerance.</td>
<td>118</td>
</tr>
<tr>
<td>5.5</td>
<td>Security pattern for the enforcement of SSL in BPMS.</td>
<td>119</td>
</tr>
<tr>
<td>5.6</td>
<td>Description of features for SSL/TLS configuration</td>
<td>120</td>
</tr>
<tr>
<td>5.7</td>
<td>Description of attributes for feature model of SSL/TLS</td>
<td>123</td>
</tr>
<tr>
<td>5.8</td>
<td>Security pattern for the enforcement of Authorization in BPMS.</td>
<td>124</td>
</tr>
<tr>
<td>5.9</td>
<td>Description of features for WAF configuration.</td>
<td>125</td>
</tr>
<tr>
<td>5.10</td>
<td>Results of feature-oriented domain analysis.</td>
<td>127</td>
</tr>
<tr>
<td>5.11</td>
<td>Analysis of generation of configurations with attributes.</td>
<td>130</td>
</tr>
<tr>
<td>5.12</td>
<td>Configurations of SSL/TLS ($SC^{c,at}$) for optimization of $ALE$ and $Cost$.</td>
<td>131</td>
</tr>
<tr>
<td>5.13</td>
<td>Configurations of fault tolerance for optimization of MTTR and Risk Reduction</td>
<td>132</td>
</tr>
<tr>
<td>5.14</td>
<td>Comparison of risk treatment selection approaches in BPM.</td>
<td>133</td>
</tr>
<tr>
<td>6.1</td>
<td>Comparison of fault-tolerance approaches for business processes.</td>
<td>154</td>
</tr>
<tr>
<td>D.1</td>
<td>Summarization of feature related to SSL</td>
<td>194</td>
</tr>
<tr>
<td>D.2</td>
<td>Summarization of feature related to TLS</td>
<td>195</td>
</tr>
</tbody>
</table>
Part I
Preface
Chapter 1

Introduction

We can only see a short distance ahead, but we can see plenty there that needs to be done.

Alan Turing

In the last decade, Business Process Management (hereinafter BPM) [Weske, 2007] has emerged in the IT industry as a solution for the enhancement of the efficiency of process management by means of automation, integration and externalization of services. In addition, there is currently a growing trend in the BPM arena towards the externalization of business processes by means of the concept of Business Processes as a Service (BPaaS) [Hurwitz et al., 2012]. Business processes have been conceived as services to be integrated with other services (third-party services, legacy systems, internal systems, etc.) in order to provide more adaptable, flexible and complex services. In Figure 1.1, an overview of the business process landscape is given.

Business processes and IT infrastructure have become a cornerstone for the management of organizations. Nevertheless, the exposure of business processes to uncontrolled elements has increased security risks. In order to ensure the development of business processes, it is crucial to deliver business processes as dependable as possible, since the materialization of problems could produce irreversible consequences for organizations along with third parties. For instance, an undetected software security fault [Marks, 2012] produced a loss of US $440 million.

Traditionally, security problems have been delegated to technical problems. However, certain scandals have demonstrated [Berger et al., 2005] that organizations need a holistic view of security, which in turn involves a wide spectrum of concerns. In general, security is often delegated to integrity, confidentiality and availability concerns. A wide overview of security can be observed from the prism of dependability. Dependability [Avizienis et al., 2004] covers integrity, confidentiality, reliability, safety, and maintainability concerns.

It is therefore essential to provide methods, techniques and tools that enhance the security awareness in the development of business processes. Nevertheless, in the majority of cases, security is considered in second stages, and security concerns in systems are often overlooked by default. Furthermore, development must comply with an enormous num-
Figure 1.1 – Business process management landscape (Weske 2007).
ber of regulations, legislations, and standards. In the current changing and competitive market, it is therefore important to ensure the traceability of security in the development process to enhance the quality of business processes.

The main aim of this thesis dissertation is the enhancement of the life-cycle of business processes by providing automatic techniques that improve the dependability of business processes.

1.1 Context and Motivation

Business Process Management (BPM) \cite{Weske2007} has emerged as an approach that includes concepts, methods, and techniques to support the design, administration, configuration, enactment, and analysis of business processes. BPM provides a development process for the implementation of operational business processes. BPM defines a life-cycle that is composed of four phases (as shown in Figure \ref{fig:1}). In the Design & Analysis phase, business processes are modelled, and validation and verification are included by simulating the execution of the just-designed processes; they are then implemented in the Configuration phase by configuring a process-aware information system (e.g., a Business Process Management System (BPMS)) which entails the testing of the implementation. In the Enactment phase, the business processes are deployed using the configured system, and the execution of process instances takes place. Finally, in the Evaluation phase, the processes are monitored and analysed to determine potential problems and to ensure the selection of appropriate repair actions.

During this life-cycle, a business process goes through various stages and systems from design to execution. Business processes are enacted using systems, such as a Business Process Management System (BPMS), and processes are executed. A deviation in the correct behaviour of business processes could produce negative effects and irreversible consequences for the organization. BPM defines tasks within life-cycle stages for the validation, verification, simulation, and monitoring of business processes. Most techniques applied in the literature are focused on applying validation and verification mechanisms for the detection of faults in the control-flow and data-flow. Nevertheless, business processes becoming ever-more threatened by security risks due to exposure of business processes to external and third parties which remain outside the control of the organizations. These security risks have to be managed since they could impede the achievement of business process goals and result in non-compliance of regulations and laws for the organization.

Nowadays, there exist a growing number of legislations and regulations that enforce conformance management of risk concerns within organizations. On the one hand, there are regulations, such as Sarbanes-Oxley (SOX) \cite{Berger2005}, EuroSOX \cite{Sorensen2011}, Basel III \cite{Basel2010}, which recommend paying attention in general-risk concerns so that they can be controlled. On the other hand, there are regulations and laws, such as the Spanish National Security Scheme \cite{Gobierno2010}, LOPD \cite{Gobierno2007}, and HIPAA, which require security-risk concerns to conform to certain criteria.

Awareness is therefore crucial during business process development regarding the con-
CHAPTER 1. INTRODUCTION

Figure 1.2 – Lifecycle of Business Process Management.

formance management of security risk concerns that helps ensure the compliance to regulations and laws. In (van der Aalst 2012), it is stated that security must be considered as orthogonal among key concerns within BPM discipline. Nevertheless, there is a lack of any comprehensive framework for the automatic management and assurance of conformance to these requirements throughout all the phases of the business process life-cycle.

This new scenario has given rise to Enterprise Risk Management (hereinafter ERM) in organizations. ERM (Anderson 2007) provides methods and processes to manage risks related to the achievement of business objectives. That is, ERM strives to provide a framework for risk management. Risk Management (hereinafter RM) techniques are often used to analyse and control security risks (Best Management Practice 2012). RM is becoming more regularized and standardized, such as that shown in ISO/IEC (SSE-CMM©) 21827:2008 and ISO/IEC 27005:2011. RM is applied in many contexts (Best Management Practice 2012) such as economy, finance, project management, and security.

The aim of these standards is to facilitate and increase the maturity of the security within the organization by means of ensuring the conformance with regard to risk objectives. RM recommends security strategies; thus controls and compliance objectives, to ensure that risks are mitigated and/or reduced with regard to the organizational needs. The integration of RM capabilities within the BPM life-cycle could enhance the quality of business processes by means of ensuring the security maturity level of business process checking the conformance of risks. Nevertheless, BPM and risk management are typically considered as disjoint concerns as stated in (Suriadi et al. 2012). Thus, there exists a
1.2. PROBLEM STATEMENTS

In the majority of cases, BPM is carried out with no regard for business. That is, business processes are designed regardless of standards, legislation compliance, and risks. Certain approaches enable the impact of risk to be reflected in the business processes as defined in the Business Motivation Model (BMM) standard by OMG. BMM provides a schema to represent the impact of potential risks in the implementation of business plans. An organization can achieve its goals by means of the implementation of business processes that follow a business policy influenced by potential risks. Nevertheless, no BMM is employed in practice together with BPM.

The integration of BPM and RM disciplines is an innovative research topic that has thrown many challenges into the BPM arena. This thesis dissertation intends to response to some of the challenges considered in these areas. This thesis dissertation endeavours a framework that enables the dependability in business process development to be ensured. One of the foundational building blocks of our framework is a conceptual model that supports both the conformance management of security concerns and its relation to business processes. The conceptual model attempts to capture and manage requirements and their relationship to business processes in a transparent and verifiable manner. When the target requirements are not achieved, the framework also incorporates a set of security patterns to facilitate the specification of countermeasures that can be used to automate the conformance achievement. Moreover, the framework is equipped with fault-tolerance capabilities in order to ensure that uncontrolled security risks cannot distract from achieving of the conformance of the risks. A set of integrated tools have been developed in order to support the framework and have been successfully validated in several case studies involving industry companies. These tools has supported the ISO/IEC 27001:2007 certification process of security in business processes of the R+D projects developed.

In the next section, we detail the problem statements addressed in this thesis.

1.2 Problem Statements

This thesis seeks the enhancement of the BPM life-cycle by providing automatic techniques that improve the dependability by means of risk management and fault tolerance of business processes as shown in Figure 1.3. This thesis dissertation presents an enhanced life-cycle in the design and run-time phases of business processes.

This challenge can be addressed by considering the following statements:

Risk-aware and Conformance of Risks Business Processes

Business process modelling constitutes an essential and crucial task in the BPM life-cycle. Business process models gather a large amount of information and structures (e.g. activities, data objects, and participants), which are vulnerable and threatened. Nevertheless, there is insufficient information to carry out a risk assessment. Business process models need to be enhanced with information that enables the risk
CHAPTER 1. INTRODUCTION

Figure 1.3 – Enhanced Business Process Management.

analysis (e.g. that of threats, vulnerabilities, countermeasures, etc.) Several approaches deal with the enhancement of business process languages to aid risk assessment, such as Lambert et al. (2006). Nevertheless, these approaches, regardless of the assessment, conform to objectives, such as time and cost in BPM. Typically, extensions to business process models are provided; however, there is an absence of formalism and supporting tools for these extensions (van der Aalst, 2012). Other approaches present overcomplicated graphical extensions, such as that given in Cope et al. (2010). Furthermore, most approaches in the literature fail to provide verification mechanisms/tools to automate the checking conformance of business processes with regard to certain objectives.

Selection and Generation of Risk Treatment for Business Processes

Risk detected within business processes must be reflected during the implementation. Risk treatment involves the task of selecting adequate countermeasures that mitigate risks. Treatments can vary from abstract to specific countermeasures depending on the level of specification (Symantec, 2009). This heterogeneity renders not only the modelling a highly complex task, but also the unified and automatic selection of countermeasures. Furthermore, the selection and configuration of security countermeasures remains one of the main problems within the scope of IT security since, in most cases it is a human, manual, time-consuming and error-prone task that involves several security stakeholders, such as security managers and administrators (Davis, 2012). There is a lack of inference mechanisms that enable security countermeasures to be selected on the basis of appropri-
1.3. THESIS CONTEXT AND PUBLISHED RESULTS

Therefore, it is essential to provide business and security stakeholders, involved in the development of business processes, with tools that aid and support the inference of security countermeasures, in accordance with those risks that need to be treated and with objectives of the organization.

Fault tolerance in the execution of business processes

Business processes are enacted and executed in systems and interact with services in a heterogeneous environment. Despite risk treatment plans and countermeasures, business processes remain non-free of faults since faults, such as zero-day attack, vulnerabilities, and failures in third party services may occur. Typically, business processes are analysed in order to detect design faults in the control-flow (Sadiq et al. 2000): bottlenecks, deadlocks, etc. On the other hand, other approaches strive to encounter faults in the data-flow (Borrego et al. 2009).

It is therefore crucial to provide business processes with fault-tolerance infrastructures that ensure the correct execution of business processes in spite of faults. Although executable business process languages are capable of implementing fault-tolerance patterns (Dobson 2006), the implementation of fault tolerance requires of an infrastructure for error detection and recovery mechanisms (Avizienis et al. 2004). Error detection is responsible for checking the behaviour of business processes (data inputs and outputs) and for diagnosing the points at fault. Subsequently, error detection should capable of triggering a recovery mechanism that corrects said fault.

1.3 Thesis Context and Published Results

This thesis dissertation has been developed in the context of the Quivir Research Group of the University of Seville under the scope of the following research projects:

- **OPBUS**: Improving business process quality by means of optimized and fault-tolerance technologies (P08-TIC-04095) funded by the Department of Innovation, Science and Enterprise of the Regional Government of Andalusia.

- **TDiaCO-BPMS**: Techniques for diagnosis, reliability and optimization in business process management systems (TIN2009-13714) funded by Spanish Ministry of Science and Technology.

- **Automated fault detection, diagnosis and tolerance in uncertain systems and distributed systems** (DPI2006-15476-C02-00) funded by Spanish Ministry of Science and Technology and ERFD/FEDER.

The following papers have been published as either intermediate or directly results of the research findings presented in this thesis:

• Angel Jesus Varela-Vaca, and Rafael M. Gasca, *Propuesta para la generación y selección adaptable de configuraciones de seguridad para sistemas de gestión de procesos de negocio*. 1er Workshop de Investigación en Tecnologías de Seguridad TIC. León, Spain, 2012.


1.3. THESIS CONTEXT AND PUBLISHED RESULTS


The following paper has been submitted to a relevant journal as direct results of the research findings presented in this thesis:


The author has also participated actively in other relevant contributions during the development of this thesis:


- Luisa Parody, María Teresa Gómez-López, Rafael M. Gasca and Angel Jesus Varela-Vaca, **Improvement of Optimization Agreements in Business Processes Involving Web Services**, Volume 2012 (2012), Article ID 959796, Communications of the IBIMA, 15 pages DOI: 10.5171/2012.959796


The research and tools developed in this thesis have been successfully applied in the process and attained the certification of the Quivir Research Group:
• ISO/IEC 27001:2007 (Registration number: ES-S1-0001/2012) by IQNet and AENOR.

Currently, the research tools developed in this thesis are being extended as part of the research of other researchers in the context of the Quivir Group.

1.4 Roadmap: Structure of the Thesis

The structure of this thesis is illustrated as a business process diagram such as drawn in Figure 1.4. The thesis dissertation encompasses four main parts:

Part I: Preface. In this section, the research context, motivation, problem statements and research findings of the thesis are introduced. To conclude this part, a list of the most relevant results attained during the development of the thesis are presented.

Part II: Background. In this section, we introduce to the reader to the most relevant concepts in the different areas in which the thesis has been developed. In Chapter 2.1 we introduce the main concepts of Business Process Management (BPM) and focus on the presentation of aspects referring to the life-cycle and business process modelling. In Chapter 2.2 we introduce the main concepts and definitions related to Risk Management. Chapter 2.3 gives an introduction of the most commonly used methodologies regarding fault diagnosis by means of model-based diagnosis. In Chapter 2.4 we present the main concepts and definitions regarding Constraint

![Figure 1.4](image-url)
Programming. Constraint Programming has been used as a cornerstone in the implementation, such as the automation of diagnosis. In Chapter 2.5, an introduction to Feature-Oriented Model Analysis is given and together with an explanation of how feature models are formalized using constraint satisfaction problems to carry out feature model analysis. In Chapter 2.6, the basis of dependability and an introduction to the most relevant fault tolerance techniques is given.

Part III: Contributions. This section is the core of the thesis which is comprised of four chapters. The chapters are structured in the following sections: a introduction to the context and motivation for the contribution; a body where the proposal solutions are presented; the results obtained addressed by examples and case of studies; a related work section where a literature review and comparisons with other approaches are given; and to conclude, a summary with a discussion of the results. In Chapter 3, we present OPBUS; the proposed framework under which the research findings are addressed. In Chapter 4, we present a contribution for the automatic risk assessment of business processes at the design phase, providing a automatic diagnosis of non-conformance of risks for business process models. Chapter 5 presents a contribution which proposes the selection and generation of optimal countermeasures for business processes, providing automatic techniques based on feature model analysis that allow the selection and generation of security configurations using multi-objective searches. In Chapter 6, a contribution for the fault tolerance of business processes is given. This contribution presents a fault tolerance approach for executable business processes by developing error detection enhanced with fault diagnosis techniques, and by reacting recovery mechanisms based on replication, redundancy, roll-back, check-pointing and diversity approaches.

Part IV: Conclusions and Future Work. This part concludes the thesis over two chapters: Chapter 7, which presents a global summary of the main conclusions that were obtained during the thesis; and Chapter 8, which outlines research lines and topics for future work that may be addressed.

Part V: Appendices. This part presents the annexes generated as a complement of the information given in the various chapters of the thesis.
Part II

Background
Chapter 2
Foundations

You take the blue pill, the story ends, you wake up in your bed and believe whatever you want to believe. You take the red pill, you stay in Wonderland, and I show you how deep the rabbit hole goes.

Morpheo. The Matrix.

Since the main aim of this thesis dissertation is the quality improvement of the BPM by means of risk management, diagnosis and fault tolerance techniques. This chapter provides the basis and background regarding BPM, risk management, diagnosis, constraint programming, feature model analysis, dependability and fault tolerance.

2.1 Business Process Concepts

There exist a growing trend to automate and externalize of services. According to this trend business processes has been build by means of the concept of Business Process as a Service (BPaaS) (Hurwitz et al., 2012). Organizations must ensure a high level of quality in their business process (cd. Definition 2.1.1).

Definition 2.1.1. A Business Process (BP) consists of a set of activities that are performed in coordination in an organizational and technical environment. These activities jointly realize a business goal (Weske, 2007).

The quality of business processes can be measured using different dimensions such as effectiveness, security, safety and reliability (Heravizadeh et al., 2009). In general, analysis techniques such as verification, validation, and diagnosis are applied to improve the dimensions of quality of business processes as in (Borrego, 2012).

In recent years, Business Process Management (BPM) (cf. Definition 2.1.2) has evolved as keystone of in the IT industry. BPM has emerged as an evolution of the traditional Workflow Management (WfM) (van der Aalst, 2004). BPM is continuously evolving in order to improve the quality and efficiency of business processes.
Definition 2.1.2. Business Process Management (BPM) is a approach that includes concepts, methods, and techniques to support the design, administration, configuration, enactment, and analysis of business processes Weske (2007) (van der Aalst et al., 2003).

BPM is orchestrated through a life-cycle such as shown in Fig. 1.2. The life-cycle of BPM to support BPs has four phases: in the design phase, the BPs are designed or redesigned; then, they are implemented in the configuration phase by configuring a process-aware information system (cf. Definition 2.1.3); in the enactment phase, the BPs are executed and deployed using the configured system; finally, in the evaluation and diagnosis phase, the processes are monitored and analysed to determine possible problems and to ensure the selection of appropriate improvement actions.

Definition 2.1.3. A Business Process Management System (BPMS) is a generic software system that is driven by explicit process representations to coordinate the enactment of business processes Weske (2007).

BPM life-cycle is focused on the design of business process models, and next diagnosis of errors in the execution of these business processes. When a business process is monitored, some errors can be detected. Nevertheless, most of these errors are related to the structure of the business process model Huang et al. (2008) (i.e. activities that cannot be achieved because of conditions that are defined incorrectly) and in the behaviour of the business process Borrego et al. (2010a) (i.e. identifying activities responsible of the incorrect behaviour). Security issues are completely considered in a second thought of BPM. Since security is consider orthogonal issues between the BPM concerns van der Aalst (2012). Nevertheless, the cost and consequences of the materialization of threats it could range from mildly annoying to catastrophic, with serious injury occurring or systems destroyed, reputation losses, security breaches, deviation of the correct service behaviour and service delivering, etc.

2.1.1 Business Process Modelling

Business process modelling remains an important factor in BPM van der Aalst (2012). Typically, processes are specified in an imperative way, i.e., explicitly specifying all possible sequences of activities in a process. However, declarative process models, i.e., implicitly specifying the allowed behaviour of the process with rules that must be followed during execution, have been increasingly used. In a declarative model (e.g., constraint-based model) everything that is not forbidden is allowed. Declarative languages based on LTL (Linear Temporal Logic), e.g., ConDec van der Aalst and Pesic (2006), can be fruitfully applied in the context of compliance checking Burattin et al. (2012). Although, LTL constraints are not easily readable for non-experts.

The most widely known imperative modelling techniques are: Flowcharts, Petri Nets, Event-driven Process Chain (EPC), UML Activity Diagram, Data Flow Diagram (DFD), Business Process Management Notation (BPMN), and IDEF3. Currently, BPMN 2.0 is the standard by OMG and it is the most used language in the market for the business process modelling. The selection of an adequate graphical method has become an important issue for both academic researchers and business professionals, since each individual
2.1. BUSINESS PROCESS CONCEPTS

Figure 2.1 – Example of business process model in BPMN notation.

process modelling method have its own characteristics. As a consequence, there are many research efforts dedicated to comparing and transforming these process modelling methods. In (Huang et al., 2008), a comparison of these major graphical process modelling methods is presented. Extensive literature research regarding another type of business process compliance has been presented (Sadiq et al., 2007) (Namiri and Stojanovic, 2007).

According to (Weske, 2007), there are two main perspectives in the development of business process in BPM: (1) Operational business process which defines the activities and their relationships, but implementation aspects of the business process are overlooked; (2) Implemented business process which retains information of the execution of activities, technical, and organizational environment in which they will be executed.

Operational business processes are specified by business process models (cf. Definition 2.1.4).

Definition 2.1.4. A business process model consists of a set of activities and execution constraints between them (Weske, 2007).

In general, business process models are composed of two perspectives: control-flow and data-flow.

Definition 2.1.5. Control-flow refers to the order in which activities are performed within a business process model.

According to BPMN 2.0 specification, the control-flow is defined by gateways, events and sequence flows (Object Management Group (OMG) 2011). In Figure 2.1 there is a description of basic elements of a business process model in BPMN notation. BPMN specification enables to define basic and complex control-flow patterns such as indicated in (Russell et al., 2006). In Figure 2.1 a Parallel split and Synchronization pattern are shown.

Another important aspect of business process models is the data-flow perspective (cf. Definition 2.1.6).
**Definition 2.1.6.** Data-flow is the data used and consumed during the execution of a business process.

In business process modelling the data-flow is described by data object artifacts attached to a task or/and sequence flows. Figure 2.1 shows an artifact attached to a task. In general, there are two types of data objects: inputs and outputs.

Several analysis techniques are used for the identification of faults or errors in both control-flow and data-flow perspectives. The majority of works in the literature are focused on the identification of structural faults in business process models. In (Huang et al., 2008), the authors strive to identify activities that cannot be achieved because of conditions that are defined incorrectly. In (van Dongen et al., 2006), the authors propose an approach for checking the soundness of business process models using a pattern-based error detection. Regarding the behaviour of business processes from a data-flow perspective; in (Borrego et al., 2010a), the authors strive to identify activities responsible for the incorrect behaviour. On the other hand, Enactment and Evaluation phases strive to identify errors in the execution of business processes.

### 2.1.2 Business Process Analysis

The analysis of business process models has become a key concern within BPM discipline (van der Aalst, 2012). Since the current thesis dissertation aims the quality improvement of business processes, business process analysis is a key concern to consider. Business Process Analysis can be defined as given in Definition 2.1.7.

**Definition 2.1.7.** Business Process Analysis (BPA) is the activity of reviewing business processes at different stages of their life-cycle in order to ascertain how far they achieve the business objectives, ranging from model verification at design time to the monitoring of processes at run-time (Borrego, 2012).

There exist a vast number of BPA approaches the most relevant analysis techniques are summarized in the following categories:

- **Verification analysis methods** (Borrego, 2012), which entails methods to detect syntax errors or violations in the control-flow and data-flow perspectives of designed business processes. In general, this kind of analysis methods are performed at design time, in order to correct these errors, avoiding the incorrect modelling of business processes. Examples of such errors are (1) common control-flow anomalies (deadlock, livelock, etc.); (2) the data-flow anomalies concerning, for example, read/write conflicts.

- **Validation analysis methods** (Mendling, 2009), which addresses the consistency of the model with the universe of discourse. In general, validation methods are used to find semantic or logical conflicts in the models.

- **Diagnosis methods** (Borrego, 2012), in order to determine the activity or activities which are the cause of a malfunction when, after the execution of a process instance, the behaviour of the business process does not correspond to the expected.
2.2. RISK MANAGEMENT

According to the literature review [Weske (2007), Huang et al. (2008), van der Aalst (2007)], BPA methods can be classified with regard to the aspect of the business process considered and the stage of the life-cycle where it is applied: at design-time and run-time.

In the revised literature, there exist an absence of BPA approaches that take into consideration the analysis and evaluation of risks in business processes. [Conforti et al. (2011)] states that BPM and risk management disciplines are largely disjoint and operate independently of one another. In recent years, there has only emerged few approaches [Suriadi et al. (2012)] in academia that provide a risk-aware business process management. Nevertheless, [Suriadi et al. (2012)] states the main research gaps to deal with BPM and risk management are: (1) the degree of formalization of various manifestations of risk within a process model is minimal - most approaches formalize their risk-related constructs at the syntactic level only (without execution semantics), thus limiting the ability to formally perform and, (2) there is a lack of research to support risk-aware business process design, and (3) the adoption of existing risk management techniques and standards into BPM systems should be strengthened.

Since there exist evident gaps in the literature that deal with BPM and risk management and the current thesis dissertation proposes an approach to the quality improvement of BPM by means of risk management, in next sections most relevant concepts and the basis of risk management are detailed.

2.2 Risk Management

The integration of BPM and risk management is between the strategic objectives defined by the European Network and Information Security Agency (ENISA) [European Network and Information Security Agency (2012), and European Commission in the ICT Trust and Security section [European Commission 7th Framework Programme (2012)]:

- ENISA: "Integration of RM/RA with Operation Processes".
- ICT Trust and Security section: "1.4 Secure, dependable and trusted infrastructures"

Risk management is defined by ISO/IEC 31000:2009 as follows. A risk management process allows the identification of assets, the appraisal and treatment of potential vulnerabilities and threats.

**Definition 2.2.1.** Risk management (RM) is a process that systematically applies management policies, procedures, and practices to a set of activities intended to establish the context, communicate and consult with stakeholders, and identify, analyse, evaluate, treat, monitor, and review risks.

RM has been widely applied to many areas (Hopkin 2010) such as corporate finance, Enterprise Risk Management (ERM), security, project management, etc. Recently, IT risk management has been emerged within ERM which refers to the risk related to Information Technologies (IT). A large number of methodologies has been developed to deal with IT risk management such as MAGERIT, CRAMM, OCTAVE, ISRAM, CORAS, NIST SP
Figure 2.2 – Overview of risk Management life-cycle.

In general, RM is addressed by a life-cycle such as shown in Figure 2.2. In general, *Establish context*, where the assets, threats and vulnerabilities are identified. Subsequently, *Risk assessment* is carried out where assets are evaluated in order to determine risks, in order to act on those risks. *Risk Treatment* where countermeasures are selected and finally there is a monitoring stage. *Risk Monitoring* stage where the effectiveness of countermeasures are continuously checked as a feedback process in order to re-evaluate the RM process in the case that any risks are not well treated.

Next definitions are based on the standards ISO/IEC 73:2009 and ISO/IEC 27005:2001 that provides a vocabulary for RM and guidelines for information security RM.

Risk is the key concept in RM that is defined as follows:

**Definition 2.2.2.** A *risk* is defined as the effects of uncertainty on the achievement of the goals.

In general, the uncertainty determines the deviation of goals due to the impact and consequence produced by potential events. On the other hand, goals refers to different aspects as security, financial, health, environmental, etc. In different levels of an organization such as project, product, processes, and entire organization.

To this end, an organization has to define the context (cf. Definition 2.2.3) for the risk assessment and to establish a risk criteria (cf. Definition 2.2.3). Spanish legislation referred to the data protection (LOPD) establishes a scale of security levels (High, Medium, Low) with regard to the data stored for an organization. For instance, an organization decides to establish a risk criteria that risks must not exceed the Medium scale of security. A risk criteria could be the acceptance risk threshold.

**Definition 2.2.3.** *Context* is the environmental internal and external factors used to achieve goals of the organization.

**Definition 2.2.4.** A *risk criteria* is a term of reference used to evaluate the importance of a risk.
2.2. RISK MANAGEMENT

2.2.1 Risk Assessment

According to ISO/IEC 27005:2011 standard, the risk assessment (cf. Definition 2.2.5) is a process which encompass the identification, estimation and evaluation of risks.

Definition 2.2.5. Risk assessment is a process in which risks are analysed and evaluated.

Definition 2.2.6. Risk analysis is a process that enables to understand the risk nature and to determine the level of risk.

In general, risk analysis is divided into two processes:

1. Risk identification is a process which identifies source of risks (that is vulnerabilities and threats), their probability of occurrence and potential impact or consequence.

2. Risk estimation is the process of computation of a list of risks with value levels are assigned.

Definition 2.2.7. Risk evaluation is a comparison process in which the results of risk analysis and risk criteria are compared to determine whether risks and its magnitude are acceptable or admissible.

There are other definitions related to the risk evaluation such as risk appetite, risk aggregation, risk tolerance that can be consulted in the glossary provided in Section 2.7.

2.2.2 Risk Treatment


Definition 2.2.8. Risk treatment is a process of selection and implementation of controls (cf. Definition 5.2.2) which intends to modify the risks.

Definition 2.2.9. Control/measure/countermeasure is something that modify a risk.

Risk treatments can be classified as follows:

- Risk avoidance, which indicates a decision not to become involved in, or action to withdraw from, a risk situation.

- Risk retention, which indicates the acceptance of the burden of loss, or benefit of gain, from a particular risk.

- Risk transfer, which indicates a sharing with another party the burden of loss or benefit of gain, for a risk.

- Risk reduction, which indicates the limitation of any negative consequence of a particular event.
Risk treatments could be insufficient to reduce the entire risk level, in this case emerged the concept of residual risk (cf. Definition 2.2.10).

**Definition 2.2.10.** *Residual risk* indicates the risk left over after the implementation of a risk treatment.

Risk management is a never-ending cycle which in last stage monitor that treatments work as planned.

### 2.2.3 Risk Monitoring

It is crucial to control the effectiveness and adequateness of risk treatments with regard to the risk criteria. To this end, a review process through risk monitoring (cf. Definition 2.2.11) is used to supervise and continually check the risks.

**Definition 2.2.11.** *Risk monitoring* is a process of verification, monitoring and critical observation or determination of the state to identify a continuously changes that may arise in the performance or intended level.

In the revised literature there exist several approaches deal which with the extension of business process models to aid and to support risk analysis. Nevertheless, (Suriadi et al., 2012) state that there is an absence of formalisms, a limited ability to formally perform, a lack of supporting risk-aware business process model design, and an absence of the adoption of existing risk management techniques and standards into BPM systems.

As stated in previous chapters, the current thesis dissertation aims to bring risk-awareness to BPM by providing a flexible, agile and automatic way for the risk assessment in the design of business processes. We propose the definition of tools in order to support the performance of an automatic risk evaluation of business processes, by taking into consideration multiple risk management methods and standards. These tools enable the automatic checking of conformance of risks of business processes with regard to business objectives related to risks. It is very important to identify which parts within a business process are non-conformance with risk objectives. To this end, fault diagnosis techniques are proposed to perform the identification of non-conformance of risks in business processes. For a better understanding of fault diagnosis, the basis of model-based diagnosis is detailed in the following section.

### 2.3 Model-based Fault Diagnosis

The aim of fault diagnosis (cf. Definition 4.3.3) is to improve the reliability, security, efficiency, and maintainability of any system. A fault diagnosis system is used to detect, isolate, and determine the location of a fault in a monitored system.

**Definition 2.3.1.** *Fault diagnosis* is a method which determines why a correctly designed system fails to work as expected.
2.3. MODEL-BASED FAULT DIAGNOSIS

The monitoring should be a reflection of the real behaviour of the system and the produced deviations from the expected behaviour. The diagnosis enables the identification of the parts which fail, and determines why a correctly designed system does not work as expected. The explanation of any abnormal behaviour, from a determined observation, is the main goal of the diagnosis.

**Definition 2.3.2.** Diagnosis is a process which identifies and isolates the reason of any unexpected behaviour, or in other words, determines which parts are failing in a system.

Model-based diagnosis [Davis (1984)](#) is recognized as a very powerful tool within the community of diagnosis due to its ability to solve the problem of isolating faulty components.

In recent decades, model-based diagnosis has become the most extensive research area in the field of diagnosis. The reasoning is carried out based on a model which explicitly represents the system to be diagnosed. A fault exists when the observed behaviour does not correspond with the behaviour derived from the model. This model is founded on the knowledge of the system. The component responsible for the fault is identified through a later analysis of the discrepancies.

Model-based diagnosis is based on the comparison between the available observations on the operation of a system and the predictions made from the model of the system. The observations indicate how the system is behaving, whereas the model expresses how it should behave during a correct execution.

When a symptom is detected, that is, a discrepancy between the observed and expected behaviour is detected, it is deduced that at least one of the components involved is not working correctly. The description of the systems, performed by the models, uses the relations between inputs and outputs. Most of the approximations for components characterize the diagnosis of a system as a collection of minimal sets of components that fail to explain the observed behaviour (symptoms). Hence it is important to be able to count on a detailed model to determine the diagnosis of a system. With this kind of model, it is possible to quickly diagnose the main parts of the system.

There are two distinct and parallel research communities that work on model-based diagnosis: DX and FDI. However, there exist hybrid approaches, as proposed by [Cordier et al. (2000)](#), where the authors propose a framework that combines both communities.

For a better understanding of both approaches, a toy example is given in Figure 2.3. This is a widely used example to support the explanations from [Davis (1984)](#) and is composed of three multipliers (M1, M2, and M3) and two adders (A1, A2) referred as the *polybox example*.

This example is used in the rest of the section to clarify the explanations of the DX and FDI methodologies.

### 2.3.1 DX: Logical Diagnosis Approach

DX community [Davis (1984)](#) presents an approximation that allows the diagnosis of systems to be performed using their structure and behaviour. The works of [de Kleer et al. (1992a)](#) and [Reiter (1987)](#) introduce the basic definitions and foundations of diagnosis. In
order to explain the discrepancies between the observed and the correct behaviour, a general theory was proposed, using a logical-based diagnosis process.

For the DX community, the system model is defined as follows:

**Definition 2.3.3.** A system model (SM) is composed of the pair \( (SD, COMPS) \), where \( SD \) (system description) is a set of logic equations, and \( COMPS \) is a finite set of components. The system description employs the predicate \( AB \) to represent the abnormal behaviour of the system, and \( \neg AB(c) \), where \( c \) is a component, to represent the correct behaviour of \( c \).

For the polybox example in Figure 2.3:

- \( COMPS = \{A_1, A_2, M_1, M_2, M_3\} \)
- \( SD = \{ADD(x) \land \neg AB(x) \rightarrow Output(x) = Input1(x) + Input2(x), MULT(x) \land \neg AB(x) \rightarrow Output(x) = Input1(x) \ast Input2(x), ADD(A_1), ADD(A_2), MULT(M_1), MULT(M_2), MULT(M_3), Output(M_1) = Input1(A_1), Output(M_2) = Input2(A_1), Output(M_2) = Input1(A_2), Output(M_3) = Input2(A_2), Input2(M_1) = Input1(M_3)\} \)

The diagnosis problem lies in the discrepancy between the set of available observations and the correct behaviour of the system described in the model.

**Definition 2.3.4.** A set of observations \( OBS \) is a set of first-order predicates.
For the polybox example:

\[
\begin{align*}
\text{OBS} = \{ & \text{Input}_1(M1) = 2, \text{Input}_2(M1) = 3, \text{Input}_1(M2) = 2, \\
& \text{Input}_2(M2) = 3, \text{Input}_2(M3) = 2, \\
& \text{Output}(A1) = 10, \text{Output}(A2) = 12 \} 
\end{align*}
\]

**Definition 2.3.5.** A diagnosis problem is represented by \( \{SD, COMPS, OBS\} \), where \( \{SD, COMPS\} \) is the system model and \( OBS \) is a set of observations.

**Definition 2.3.6.** A diagnosis for \( \{SD, COMPS, OBS\} \) is a set of components \( D \subseteq COMPS \), such that \( SD \cup OBS \cup \{AB(c) | c \in D\} \cup \{\lnot AB(c) | c \in COMPS - D\} \) is satisfied. A minimal diagnosis is a diagnosis \( D \) such that \( \forall D' \subset D, D' \) is not a diagnosis.

In order to attain the minimal diagnoses, [Reiter 1987]{#Reiter} propose the concept of conflict for their generation, since this concept constitutes the base for most DX approaches.

**Definition 2.3.7.** An R-conflict for \( \{SD, COMPS, OBS\} \) is a set of components \( C = \{c_1, c_2, \ldots, c_n\} \subseteq COMPS \), such that \( SD \cup OBS \cup \{\lnot AB(c) | c \in C\} \) is inconsistent. A minimal R-conflict is an R-conflict which does not contain any other R-conflict.

**Definition 2.3.8.** A hitting set is a set of components which intersect the minimal R-conflicts. The minimal hitting set is a set that includes a component from each minimal R-conflict.

Using the concepts of minimal R-conflict and minimal hitting set, the minimal diagnosis can be formalized as follows:

**Definition 2.3.9.** \( D \) is a minimal diagnosis for \( \{SD, COMPS, OBS\} \) iff \( D \) is a minimal hitting set in the collection of minimal R-conflicts of \( \{SD, COMPS, OBS\} \).

For the polybox example in Figure 2.3 with \( f = 10 \) and \( g = 12 \), there are four minimal diagnoses given by the minimal hitting sets \( \{\{A1, A2, M1, M3\}, \{A1, M1, M2\}\} \), which are: \( D_1 = \{A1\}; D_2 = \{M1\}; D_3 = \{A2, M2\}; D_4 = \{M2, M3\} \).

### 2.3.2 FDI: Analytical Redundancy Approach

[Cassar and Staroswiecki 1997]{#Cassar} and [Staroswiecki and Declerk 1989]{#Staroswiecki} present the formalization of the process to obtain the ARR(s) (Analytical Redundancy Relation) of the system. This process is based on searching the overdetermined systems in which the detection and location of the faults is possible. The FDI methodology allows an off-line analysis of a part of the work, in contrast with the DX methodology where the work is almost completely on-line. [Fattah and Holzbaur 1994]{#Fattah} propose an approximation to the FDI methodology, but use logic constraint models to solve the diagnosis problem.

For the FDI approach, the behavioural model (BM) is derived from its structure, which shows the connections between the components and the behaviour of each component. For this reason, the FDI methodologies are useful when it comes to diagnosing business processes where only the control flow perspective of the model is available.
Definition 2.3.10. A System Model (SM) is defined as the behavioural model (BM), i.e., the set of relations of the model, together with the observation model (OM).

For the polybox example in Figure 2.3 the system model is:

- BM : \{RM1 : x = a \ast c; RM2 : y = b \ast d; RM3 : z = c \ast e; RA1 : f = x + y; RA2 : g = y + z\}

- OM : \{RSa : a = a_{obs}; RSb : b = b_{obs}; RSc : c = c_{obs}; RSD : d = d_{obs}; RSe : e = e_{obs}; Rsf : f = f_{obs}; Rsg : g = g_{obs}\}

Definition 2.3.11. A diagnosis problem is defined by a system model (SM), a set of observations (OBS), and a set of faults (F).

For the polybox example in Figure 2.3:

- OBS = \{a_{obs} = 2; b_{obs} = 2; c_{obs} = 3; d_{obs} = 3; e_{obs} = 2; f_{obs} = 10; g_{obs} = 12\}

- Set of single faults (SF): \{FA1, FA2, FM1, FM2, FM3\}, where \(F = 2^{SF}\).

Definition 2.3.12. The structure of a system is defined by means of a binary application \(S : SM \times V \rightarrow \{0, 1\}\), where \(V = X \cup O\) is the set of variables, and where \(s(\text{rel}, v) = 1\) iff \(v\) appears in the relation \(\text{rel}\).

Definition 2.3.13. An Analytical Redundancy Relation (ARR) is a relation established by the SM, that contains only observed variables and can therefore be evaluated by OBS. It should be noted that \(r = 0\), where \(r\) is the residual of the ARR. For a given OBS, the instantiation of the residual is noted by \(\text{val}(r, OBS)\), referred to as \(\text{val}(r)\), which is equal to 0 if the observations for the ARR are satisfied.

The ARRs can be obtained from the SM by eliminating the unknown variables. For the polybox example in Figure 2.3, the following two ARRs can be stated:

- ARR1 : \(r_1 = 0\), where \(r_1 \equiv f_{obs} - a_{obs} \ast c_{obs} - b_{obs} \ast d_{obs}\)

- ARR2 : \(r_2 = 0\), where \(r_2 \equiv g_{obs} - b_{obs} \ast d_{obs} - c_{obs} \ast e_{obs}\)

By assuming sensors are fault-free, these two ARRs can be rewritten as:

- ARR1 : \(f - (a \ast c + b \ast d) = 0\)

- ARR2 : \(g - (b \ast d + c \ast e) = 0\)

Additional ARRs can be deduced through the combination of the elementary ARRs. In this case, subtracting ARR2 from ARR1:

- ARR3 : \(f - g - a \ast c + c \ast e\)
2.3. MODEL-BASED FAULT DIAGNOSIS

Table 2.1 – Signature matrix for single faults

<table>
<thead>
<tr>
<th>ARR</th>
<th>$F_{A1}$</th>
<th>$F_{A2}$</th>
<th>$F_{M1}$</th>
<th>$F_{M2}$</th>
<th>$F_{M3}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

It is important to notice that the components involved in this third ARR are not a combination of the components used in the previous ARRs.

**Definition 2.3.14.** Given a set $R = \{ARR_1, ARR_2, \ldots, ARR_n\}$, composed of $n$ ARRs, and a set $F = \{F_1, \ldots, F_m\}$ with $m$ faults, the fault signature $F_j$ is given by the binary vector $FS_j = [s_{ij_1}, \ldots, s_{ijn}]^T$, where $s_{ij}$ is given by $(ARR_i, F_j) \rightarrow s_{ij} = 1$ if at least one component from $F_j$ is involved in $ARR_i$, and where $(ARR_i, F_j) \rightarrow s_{ij} = 0$ otherwise.

This specifies whether the occurrence of the fault affects the corresponding ARR.

**Definition 2.3.15.** Given a set $R$ composed of $n$ ARRs, the signature of a set of $m$ faults is called the signature matrix.

The signature matrix for single faults (SF) for the polybox example is shown in Table 2.1.

For multiple faults, it is only necessary to add the fault combinations as columns in the signature matrix.

The diagnosis of the FDI approach is based on the interpretation of the columns in the signature matrix, and lies in comparing the observation signature with the fault signature. This decision is treated as a decision-making problem.

**Definition 2.3.16.** The observation signature $OS$ is a binary vector $OS = [OS_1, \ldots, OS_n]^T$, where $OS_i = 0$ iff $val(r_{ij}, OBS) = 0$.

The first step (detection phase) lies in building the observation signature. For the polybox example in Figure 2.3, the observation signature is $OS = [1, 0, 1]^T$.

In the second step (isolation phase), the observation signature and the fault signature are compared. The result from this decision forms the definition of a fault criterion.

For the polybox example:

- $OS = [1, 0, 1]^T \leftrightarrow F_{A1}$ or $F_{M1}$ or $F_{\{A1,M1\}}$

In the case of $f = 10$ and $g = 10$:

- $OS = [1, 1, 0]^T \leftrightarrow F_{M2}$

The one aim of the current thesis dissertation is the application of Model-based Fault Diagnosis techniques for the automatic diagnosis of non-conformance of risks of business process. In the following section, the main concepts related to the Constraint Programming paradigm is presented, which is used as support for the implementation and automation of the proposed diagnosis methodologies, thanks to its efficiency and expressiveness when it comes to modelling problems which include both topological aspects and numeric data.
2.4 Constraint Programming

Constraint Programming is based on the resolution of Constraint Satisfaction Problems (CSPs), which are problems where an assignment of values to variables must be found in order to satisfy a number of constraints. A large number of problems in Artificial Intelligence and other areas of Computer Science can be seen as special cases of CSPs. Examples include scheduling, temporal reasoning, graph problems, configuration problems, etc.

Various approaches to solve these problems have been developed, a number of them which use constraint propagation to simplify the original problem. Others use backtracking to directly search for possible solutions. Several are a combination of these two techniques.

In general, a CSP (cf. Definition 2.4.1) is composed of a set of variables, a domain for each variable, and a set of constraints. Each constraint is defined over some subset of the original set of variables and limits the combinations of values that the variables in this subset can take. The goal is to find one assignment to the variables such that the assignment satisfies all the constraints. In some kind of problems, the goal is to find all such assignments [Kumar (1992)].

Definition 2.4.1. A Constraint Satisfaction Problem (CSP) consists of the triple \((V, D, C)\), where \(V\) is a set of \(n\) variables \(v_1, v_2, \ldots, v_n\) whose values are taken from finite, discrete domains \(D_{v_1}, D_{v_2}, \ldots, D_{v_n}\) respectively, and \(C\) is a set of constraints on their values. The constraint \(c_k(x_{k_1}, \ldots, x_{k_n})\) is a predicate that is defined on the Cartesian product \(D_{k_1} \times \cdots \times D_{k_n}\). This predicate is true iff the value assignment of these variables satisfies the constraint \(c_k\).

The search of solutions for a CSP is based on the instantiation concept. An assign-
2.4. CONSTRAINT PROGRAMMING

An instantiation of a variable, or instantiation, is a pair variable-value \((x, a)\) which represents the assignment of the value \(a\) to the variable \(x\). An instantiation of a set of variables is a tuple of ordered pairs, where each sorted pair \((x, a)\) assigns the value \(a\) to the variable \(x\). A tuple \(((x_1, a_1), \ldots, (x_i, a_i))\) is consistent if it satisfies all the constraints formed by variables of the tuple.

A solution to a CSP is an assignment of values to all the variables that all constraints must be satisfied. Hence a solution is a consistent tuple which contains all the variables of the problem. A partial solution is a consistent tuple which contains some of the variables of the problem. A problem is consistent if it exists, at least, a solution, i.e., a consistent tuple. In Figure [2.4] there is a graphic which represents the space of solutions of a CSP that must satisfy four constraints \((R_1, R_2, R_3, \text{and } R_4)\) therefore the space of solutions is restricted to the grey-highlighted rectangle. However, CSP must find solutions among space solutions that are restricted to an objective function. In general this solutions reduce the space of solutions to a sub-set of the solutions.

The techniques used in constraint satisfaction depend on the kind of constraints being considered. Constraints are often used on a finite domain, to the point that constraint satisfaction problems are typically identified with problems based on constraints on a finite domain. Such problems are usually solved via techniques that combines propagation and searches, in a particular form of backtracking and local searcher. Constraint propagation is another method used on such problems; the majority of them are incomplete. In general, they may solve the problem or prove it unsatisfiable, but not always. Constraint propagation methods are also used in conjunction with searches to make a given problem simpler to solve. Other considered kinds of constraints are on real or rational numbers; solving problems on these constraints is done via variable elimination or the simplex algorithm.

2.4.1 Searches

Search techniques to find solutions to a CSP are based normally on search algorithms such as backtracking or exhaustive. The try to find a solution through the space of possible assignments of values to the variables, if it exists, or to prove that the problem has not a solution. Because of this they are known as complete algorithm. The incomplete algorithms such as local searches do not guarantee to find a solution, but they are very used in optimization problems since their mayor efficiency and the high cost that a complete search requires. A lot of complete search algorithms have been developed.

2.4.2 Optimization Problems

There are often a lot of solutions to a CSP, but a user is interested only in some of them, on only in a specific one. To solve this limitation, several extensions of the model have been proposed, where it is allowed to contain weak constraints (which indicate preferences, not obligation) with different semantics, such as priorities, preferences, costs, or probabilities. In the Constraint Optimization Problems (COP) the aim is to find the best solution, where the preference criteria between the solutions is specified by the weak constraints.
and an objective function that has to be optimized.

2.4.3 Overconstrained Constraint Satisfaction Problems

When solving a CSP, it is necessary to assign values to variables satisfying a set of constraints. In real applications it often happens that problems are overconstrained and do not have any solution. In this situation, it is desirable to find the assignment that best respects the constraints under various preference criteria. Under this view, overconstrained CSPs are optimization problems for which branch and bound is a suitable solving strategy. The efficiency of branch and bound-based algorithms greatly depends on the lower bound used to detect dead ends and to avoid the exploration of large regions in the search space. This lower bound should be both as large and as cheap to compute as possible.

Approaches (Affane and Bennaceur, 1998; Freuder and Wallace, 1992; Wallace, 1995) for lower bound computation aggregate two main elements: (i) the global contribution of assigned variables, and (ii) the addition of individual contributions of unassigned variables. Another approach (G. Verfaillie and Schiex, 1996) keeps (i) but substitutes (ii) by a global contribution of unassigned variables. This is done by the Russian Doll Search (RDS) method, which requires \( n \) successive searches on nested sub-problems to finally solve a problem of \( n \) variables.

A discrete binary constraint satisfaction problem is defined by a finite set of variables \( X = \{1, \ldots, n\} \), a set of finite domains \( \{D_i\}_{i=1}^n \) and a set of binary constraints \( \{R_{ij}\} \). Each variable \( i \) takes values in its corresponding domain \( D_i \). A constraint \( R_{ij} \) is a subset of \( D_i \times D_j \) which only contains the allowed value pairs for variables \( i, j \). An assignment of values to variables is complete if it includes every variable in \( X \), otherwise it is partial. A solution for a CSP is a complete assignment satisfying every constraint. The problem is called overconstrained if such an assignment does not exist. It may be of interest to find a complete assignment that best respects all constraints (Bistarelli et al., 1995; Schiex et al., 1995). The Maximum Constraint Satisfaction Problem (Max-CSP) can be considered (Kask, 2000; Larrosa and Meseguer, 1999), for which the solution of an overconstrained CSP is a complete assignment satisfying the maximal number of constraints. That is, in a Max-CSP a number of constraints are allowed to be violated, and the quality of a solution is measured by the number of satisfied constraints. This way, in order to identify the constraints which need to be relaxed (or removed) to get a solution in an overconstrained CSP, the concept of Max-CSP can be combined with reified constraints. A reified constraint consists of a constraint associated to a boolean variable which denotes its truth value. Therefore, by maximizing the number of reified constraints whose truth values are equal to \( true \), the constraints to relax will be the rest.

Constraint Programming techniques have been used in multiple and interdisciplinary areas such as Software Engineering. In particular, Constraint Programming techniques have been used in academia for the analysis of feature models in the research scope of Software Product Line (SPL). In the third part of the current thesis dissertation, Constraint Programming techniques and Feature-Oriented Domain Analysis (FODA) have been proposed in order to optimize the reasoning and inference of selecting and generating risk treatments for business processes. In the following section, the basis of feature models
2.5 Feature-Oriented Domain Analysis

SPL (Batory, 2005; Benavides et al., 2010; Kang, 1990) is an emerging paradigm in the Software Engineering arena which provides guides for the development of products. This paradigm is based on the key idea of defining reusable components by means of the identification of core features and through product development. Current studies in SPL are focused on the domain analysis that consists of the process of analysing related products in order to identify their common and variable features. Feature model (hereinafter FM) is the most popular method for the domain analysis of a SPL. Feature-Oriented Domain Analysis (Kang, 1990) is a method to perform a domain analysis of feature models.

Feature models involve a model that defines features and their relations. FMs enable to study certain properties such as potential number of valid products, even whether a particular configuration (selection of features) constitutes a valid product. There exist various notations to design FMs (Batory, 2005), although the most widely used is that proposed by Czarnecki (Czarnecki et al., 2005) as shown in Figure 2.5. This representation enables four relations between a parent feature and its child features:

- Mandatory, child feature is required. (cf. in the figure A mandatory sub-feature of B, $A \leftrightarrow B$).

- Optional, child feature is optional (cf. in the figure A optional sub-feature of C, $A \rightarrow B$).

- Alternative, one of the sub-features must be selected (i.e in general $a_1, a_2, \ldots, a_n$ alternative sub-feature of b, $a_1 \lor a_2 \land \cdots \land a_n \leftrightarrow \bigvee_{i<j}(a_i \lor \cdots \lor a_j)$, in the figure $B \leftrightarrow D \lor E$).

- Or-relation, at least one of the sub-features must be selected (i.e in general $a_1, a_2, \ldots, a_n$ or sub-feature of b, $a_1 \land a_2 \land \cdots \land a_n \leftrightarrow b$, in the figure $C \leftrightarrow F \land G$).

In addition, other relations (cross-tree constraints) are allowed. The most common are:

- Feature A requires feature B (cf. in the figure $E \rightarrow F$).

- Feature A excludes feature B (i.e $\neg(A \land B)$).

In some cases, the expressiveness of this notation is insufficient to represent certain relations and information related to features. To overcome this drawback there exist extensions (Benavides et al., 2010; Czarnecki et al., 2005) for the inclusion of attributes and extra-functionalities for features. These extensions enable characteristics of features that can be measured to be provided and include the facility to express relations between these
characteristics (extra-functionalities). Figure 2.5 shows an example of a feature model extended with attributes for instance feature D has an attribute Z which is instantiated by the expression $Z = value1$.

Several techniques are available for the automated analysis of FMs (Benavides et al., 2010), using Propositional Logic (PL), Constraint Programming (CP), and Description Logic (DL). These approaches transform the feature models into formal models in order to infer information related to the product line. This information may include: number of products, filters (specific set of characteristics for the features), products (all products with certain features), validation (selection of characteristics represent a valid product), optimum products (determination of best products according to a set of criteria), variability (relation between number of potential products and certain products), commodity (relation between a number of certain products and the total number of products).

Constraint programming is employed to carry out feature model analysis since this approach enables integer domains to be used for attributes and optimization functions. In (Benavides et al., 2005), the authors apply a transformation to Constraint Satisfaction Optimization Problem (COP) of extended feature models which is able to automatically obtain information about the model. Figure 2.6 gives an example of transformation of a feature model to a COP whose objective function is to find the maximum of $x$.

### 2.6 Dependability and Fault Tolerance

Since the current thesis dissertation aims to provide techniques to enhance the quality of executable business processes by means of improving its dependability. This section gives an overview of the main concepts about dependability, and presents background to fault tolerance techniques.
2.6. DEPENDABILITY AND FAULT TOLERANCE

2.6.1 Basic Concepts

Nowadays, organization tend to externalize its business process as a services, and these
are integrated with other critical applications. There exist a clear need for dependable
business processes services, due to its use in critical and everyday applications, and its in-
creasing complexity. In this section, basic concepts of dependability and various technical
means to achieve dependable executable business process are introduced.

The concepts related to dependability can be classified in the form of tree as shown
in Figure 2.7. This tree represents a taxonomy for software dependability proposed by
Pullum in (Pullum, 2001). This taxonomy represents an adaptation from the classical tax-
onomy proposal by Avizienis in (Avizienis et al., 2004). The taxonomy is encompassed
of threats, means, and attributes. The means indicates the form to attain dependability.
The attributes define the properties used to measure the degree of achievement of de-
pendability. Additional attributes can be derived from the properties of those properties.
For example, security might be derived from the properties of integrity, availability, and
confidentiality.

Definitions introduced in this section are based on the taxonomies introduced to
(Abizienis et al., 2004) and (Pullum, 2001).

Definition 2.6.1. Dependability of a computing system is the ability to deliver service
that can justifiably be trusted. The service delivered by a system is its behaviour as it is perceived by its user(s); a user is another system (physical, human) that interacts with the former at the service interface. Correct service is delivered when
the service implements the specified function.

According to definition 2.6.1, dependability of a business process might be defined as follows:

Definition 2.6.2. Dependability of a business process is the ability to deliver service that
can justifiably be trusted. The service delivered by a business process is its behaviour as
it is used by its customer(s); a customer is another system (software, hardware, human)
that interacts with the former at the service interface. Correct service is delivered when
the business process implements the expected function.
The service of a business process fails when the delivered service deviates from correct service due to a failure.

**Definition 2.6.3.** A failure occurs when the service delivered by the system deviates from the specified service. This implies that the expected service is described, typically by a specification or set of requirements.

**Definition 2.6.4.** An error is that part of the system state that may cause a subsequent failure. It can be unrecognised as an error (i.e., latent) or detected. An error may propagate, that is, produce other errors.

**Definition 2.6.5.** A fault is the identified or hypothesized cause of an error. It can be viewed as simply the consequence of a failure of some other system (including the developer) that has delivered or is now delivering a service to the given system. A fault is active when it produces an error in otherwise it is a dormant fault.

Pullum’s taxonomy (Figure 2.7) define two main groups of means: (1) those used during the *construction* process of software (fault avoidance and fault tolerance); and (2) those employed during the *validation* process of software (fault forecasting and fault removal).

In general, fault tolerance strive to deliver correct service in the presence of faults. Thus, fault tolerance attempt to prevent failures by tolerating faults whose occurrences are known when errors are detected. Therefore, the means of fault tolerance applied to business process may be defined as follows:

![Figure 2.7 – Dependability taxonomy](pullum_2001)
Definition 2.6.6. Fault-tolerance is a mechanism used to guarantee the correct service of business processes by complying with the specification in spite of the presence of faults.

Fault tolerance is generally implemented by two main parts:

- *Error detection*: in which an erroneous state of the system is identified.
- *Recovery*: in which an erroneous state of the system is substitute by an error-free state.

On the one hand, there exist two classes of error detection (Avizienis et al., 2004):

(a) *concurrent error detection*, which takes place during service delivery; and

(b) *preemptive error detection*, which takes place while service delivery is suspended; it checks the system for latent errors and dormant faults.

On the other hand, recovery (Avizienis et al., 2004) involves two tasks:

(a) Error handling: eliminates errors from the system state. It may take two forms:

   i. *Roll-back*, where the state transformation consists of returning the system back to a saved state that existed prior to error detection; that saved state is a checkpoint.

   ii. *Roll-forward*, where the state without detected errors is a new state.

(b) Fault handling: prevents located faults from being activated again. It takes four steps:

   i. *Fault diagnosis* that identify cause of errors.

   ii. *Fault isolation* that performs exclusion of fault components from further participant in the service delivery.

   iii. *System reconfiguration* that reassigns tasks among non-failed components.

   iv. *System reinitialization* that checks, updates and records the new configuration and updates system tables and records.

A key aspect in the fault tolerance is the concept of *redundancy*, that is, additional resources that would be required to detect and tolerate faults. Redundancy can take several forms:

- Hardware redundancy includes replicated and supplementary hardware added to the system to support fault tolerance. It is perhaps the most common use of redundancy.

- Software redundancy includes the additional programs, modules, or objects used in the system to support fault tolerance.

- Information or data redundancy includes the use of additional information and data to assist in fault tolerance.
CHAPTER 2. FOUNDATIONS

Figure 2.8 – Basic design of diversity.

- Temporal redundancy involves the use of additional time to perform the tasks required to effect fault tolerance.

Replication is employed in the recuperation of services by means of duplication of each of its functionalities in form of replicas and in the case of a service replica fault another replica takes control. Classical solutions in hardware replication are classified as follows:

- **Passive replication** \(\text{[Liu et al., 2008]}\): clients only interact with one replica (primary) which handles the client requests and sends back responses. The primary replica also issues messages to the backup replicas (other secondary replicas) in order to update their state.

- **Active replication** \(\text{[Liu et al., 2008]}\): all replicas play the same role. All replicas receive each request, handle the request, and send back the response to the client. Other solutions based on active replication exist \(\text{[Baldoni et al., 2002]}\).

Hardware replication is an effective mechanism to tolerate hardware faults. Nevertheless, the replication it is insufficient to protect against software design and implementation faults since doing so would simply duplicate the faults into the replicas. If the same software is copied and a failure occurs in one of the software replicas, that failure will also occur in the other replicas and there will be no way to detect the problem. A solution propose in the software fault tolerance arena is the use of diversity in the replicas \(\text{[Pullum, 2001]}\); \(\text{[Torres-Pomales, 2000]}\). The basic idea is to develop the same software component with different implementations (variants). Thus, a software component starts with the same specification and have different variants independently developed. The use of diversity requires some means to adjudicate, arbitrate, or otherwise decide on the acceptability
of the results obtained by the variants as shown in Figure 2.8. In general, this component is called adjudicator.

There exist several techniques [Pullum 2001] [Torres-Pomales 2000] to design diversity at software such as N-Version Programming (NVP), Recovery Blocks (RcB), Distributed Recovery Blocks, N-Self Checking Programming, Consensus Recovery Block, Acceptance Voting, and Comparison techniques. These techniques differ in the definition of variants. For instance, NVP technique define the variants as components such as shown in Figure 2.8. Nevertheless, RcB define the variants as a checkpoint structure such as shown in Figure 2.9.

### 2.7 Glossary of terms

This section provides a alphabetical-ordered compilation of terms as a glossary for the rest of the thesis:

- **Business Process Analysis** is the activity of reviewing business processes at different stages of their life-cycle in order to ascertain how far they achieve the business
objectives, ranging from model verification at design time to the monitoring of processes at run-time.

- **Business Process Management** is a generic software system that is driven by explicit process representations to coordinate the enactment of business processes.

- **Business Process Model** consists of a set of activity models and execution constraints between them.

- **Control/countermeasure** is something that modify a risk.

- **Constraint** is a predicate composed of variables, relations between them.

- **Constraint Satisfaction Problem** consists in a set of variables, domains and constraints.

- **Constraint Programming** is a programming paradigm based on the resolution of CSPs.

- **Diagnosis** is a process which identifies and isolates the reason of any unexpected behaviour, or in other words, identify which parts are failing in a system.

- **Dependability** of a computing system is the ability to deliver service that can justifiably be trusted.

- **Error** is a part of the system state that may cause a subsequent failure. It can be unrecognised as an error (i.e., latent) or detected. An error may propagate, that is, produce other errors.

- **Fault** is the identified or hypothesized cause of an error. It can be viewed as simply the consequence of a failure of some other system (including the developer) that has delivered or is now delivering a service to the given system.

- **Failure** is a deviation of the service delivered from the specified service.

- **Fault Diagnosis** is a method which determines why a correctly designs systems fail to work as expected.

- **Fault Tolerance** is a mechanism used to guarantee the correct service of business processes by complying with the specification in spite of the presence of faults.

- **Model-based diagnosis** is a reasoning method to solve problems of isolating faulty components.

- **Residual risk** indicates the risk left over after the implementation of a risk treatment.

- **Risk** is defined as the effects of uncertainty on the achievement of the goals.

- **Risk appetite** is the amount of risk that an organization is prepared to retain.
2.7. GLOSSARY OF TERMS

- **Risk aggregation** is the combination the numbers of risks in an single risk.

- **Risk analysis** is a process that enables to understand the risk nature and to determine the level of risk.

- **Risk assessment** is a process in which risks are analysed and evaluated.

- **Risk avoidance**, which indicates a decision not to become involved in, or action to withdraw from, a risk situation.

- **Risk dimension** category of similar risks related to security dimensions.

- **Risk estimation** is the process of computation of a list of risks with value levels are assigned.

- **Risk evaluation** is a comparison process in which the results of risk analysis and risk criteria are compared to determine whether risks and its magnitude are acceptable or admissible.

- **Risk identification** is a process which identifies source of risks (that is vulnerabilities and threats), their probability of occurrence and potential impact or consequence.

- **Risk Management** is a process that systematically applies management policies, procedures, and practices to a set of activities intended to establish the context, communicate and consult with stakeholders, and identify, analyse, evaluate, treat, monitor, and review risks.

- **Risk retention**, which indicates the acceptance of the burden of loss, or benefit of gain, from a particular risk.

- **Risk transfer**, which indicates a sharing with another party the burden of loss or benefit of gain, for a risk.

- **Risk treatment** is a process of selection and implementation of controls which intends to modify the risks.

- **Risk reduction**, which indicates the limitation of any negative consequence of a particular event.

- **Risk mitigation**, which indicates the removal of any negative consequence of a particular event.

- **Risk monitoring** is a process of verification, monitoring and critical observation or determination of the state to identify a continuously changes that may arise in the performance or intended level.

- **Security dimension** is a security property related to an system.
Part III
Contributions
Chapter 3

OPBUS: OPtimization of BUsiness process Security

Imagination is more important than knowledge.
Albert Einstein.

3.1 Context and Motivation

The automation and externalization of business processes as services means that the success of an organization relies heavily on the dependability of its business process models and systems. Business analysts design business processes whereby faults in decision-making could produce irreversible consequences. Organizations must adopt business process development methods that ensure a high trust in the business processes. Business analysts must therefore pay strict attention to the early inclusion of mechanisms that promote the analysis, evaluation and treatment of the risk of security faults, spanning from the design to the execution stages of business processes.

Risk management is becoming ever-more prominent within organizations since it helps to reduce risk by means of analysis, evaluation of threats, and the definition of mitigation plans. Nevertheless, risk management has generally been applied separately from BPM with a complete lack of formalism and automation, such as stated in (Suriadi et al., 2012; van der Aalst, 2012).

On the other hand, fault tolerance has been thoroughly applied in several areas, such as distributed systems (Baroni et al., 2000), grid environments (Shi et al., 2010), and web service orchestration (Looker and Munro, 2005). Nevertheless, these approaches pay no attention to business process aspects. These type of approaches therefore strive to ensure the service delivery in the presence of faults. Most of these approaches deal with system and web service faults, such as crash faults. There exist other types of faults that are lack attention, such as unexpected behaviour due to security attacks in the web services orchestrated within a business process, and propagation of errors due to undetected errors.
in the results of a service. These types of failures require a sophisticated fault diagnosis that enable the components responsible within business processes to be determined for the activation of recovery mechanisms that ensure the propagation of correct results within a business process during the execution.

The main aim of the current thesis dissertation is to provide a framework for quality improvement of business process development by means of the enhancement of dependability of BPM from design to runtime stages. In the following sections, the framework is given in detail, and contributions articulated within framework are introduced.

3.2 Framework description

OPtimization of the BUsiness process Security (hereinafter OPBUS) framework has been developed to fill the gap between business process development, risk management and fault tolerance by means of the improvement of the BPM life-cycle. OPBUS aims to aid and support business and security experts towards improving decision-making BPM life-cycles by means of determining the most relevant risks within business process models and providing the optimal controls and fault tolerance to treat those risks.

As mentioned in Chapter 1, the business process development life-cycle is composed of at least four stages: Design and Analysis, Configuration, Enactment and Evaluation. OPBUS attempts to improve the Design and Analysis, Configuration, and Enactment phases as shown in Figure 3.1. Design and configuration phases are extended with tasks for the assessment, and treatment, of risks of business process models. Configuration and Enactment are phases whereby the implementation of executable business processes are complemented with error detection and recovery mechanisms for fault tolerance. In short, OPBUS introduces a set of contributions that can be summarized as follows:

1. Design and Analysis
   
   (a) Diagnosis of business process models identifying unexpected high level of risks.
   
   (b) Selection and generation of optimal countermeasures to mitigate risks.

2. Configuration and Enactment
   
   (a) Deployment of error detection and recovery mechanisms that ensure the delivery of correct business process services in the presence of faults at run-time.

   The diagram in Figure 3.2 shows the relation between these contributions, represented as a business process model. In accordance with the life-cycle in Figure 1.2, the input is a defined business process model. The contributions are represented as tasks within the process to provide certain functions: (1) Diagnosis of non-conformance of risks task, to perform an assessment of business process models by diagnosing non-conformances of risks with regard to expected acceptable risk levels established in objectives; (2) Selection of Optimal Configurations and Generation of Configurations tasks, using diagnosed risks within the model to perform an optimized search of countermeasures that enable configurations that reduce and mitigate risks to be generated according to the needs of the
organization; and (3) Deploy Fault Tolerance Infrastructure task, to provide implementation of business process models with error detection and recovery mechanisms. These contributions are given in detail in the following subsections.

**Diagnosis of non-conformance of risks**

As stated earlier, during the Design and Analysis phase it is crucial to carry out a diagnosis of potential risks of business process models before implementing the Configuration stage. To this end, our framework relies on the Diagnosis of non-conformance of risks which is in charge of providing the diagnosis of non-conformance of risks of business process models with regard to the objectives related to risks.

Business process models gather a large amount of information and structures (e.g. activities, data objects, participants, and so on) that are susceptible to being vulnerable and threatened. Nevertheless, business process models need to be extended with information that enables the assessment of business process models with regard to risks (e.g. threats, vulnerabilities, and countermeasures). Our proposal strives to convert business process models onto risk-aware business process models by providing a risk model as an extension of business process models.

For the Business Process Modelling stage, Validation and Verification (V&V) methods have been fully employed by other research in order to detect and diagnose faults. In general, these methods search for design faults and unexpected behaviour such as bottlenecks,
deadlocks, lack of synchronization, missing data, and redundant data. Thus, these are used to detect faults from control flow and data flow perspectives. Other types of faults, such as those produced by security threats and vulnerabilities, remain outside the scope the aims of these approaches.

Our proposal provides a contribution focused on the provisioning of verification methods for the risk assessment of business process models, and for the diagnosis of risks that are in non-conformance with regard to a risk level objective. More precisely, our proposal provides:

• techniques for the automatic risk assessment of business process models regarding the risk information and the control flow perspective, and

• the consequent automatic diagnosis for the conformance checking of business process models with regard to objectives such as an acceptable risk level.

This contribution therefore provides a fully automated approach in Chapter 4 for the assessment and diagnosis of risks in business process models. To this end, this contribution uses a modelling of business process models (control flow perspective and a risk extension) in a Constraint Programming (CP) paradigm and Model-Based Diagnosis (MDB) principles to check the conformance of the model with regard to the established objectives.

Selection and Generation of configurations

As aforementioned, risk treatment is the process in charge of selecting and implementing adequate countermeasures that, in accordance with Chapter 2, reduce, retain, transfer, or prevent risks. In particular, we are interested in controls that produce a risk reduction. In general, risk treatments are stated in a mitigation plan composed of a set of countermeasures. Nevertheless, treatments can vary from abstract to specific countermeasures.
depending on the level of specification. This heterogeneity renders the modelling highly complex for the selection of countermeasures in unified and automatic way. Moreover, it is crucial for organizations to ensure that selected countermeasures respond to business objectives. On the other hand, security countermeasures are mostly stated by software and hardware configurations. It is crucial to ensure that configurations referring to the countermeasures are generated during the implementation of business process models as a response to risks.

Selection of optimal configuration and Generation of configurations contributions are in charge of the selection, inference, and generation of optimal countermeasures, detailed in Chapter [5]. To this end, the contribution firstly provides a formalization based on security patterns and feature models for the representation of countermeasures. Furthermore, the contribution provides automatic techniques based on Feature-Oriented Domain Analysis (FODA) and Constraint Programming for the inference, selection and generation of optimal configurations with regard to single and multiple objectives.

Deployment fault tolerance infrastructures

In spite of countermeasures, executable business processes remain non-free of faults (e.g. zero-day threats). Therefore, it is crucial to provide executable business processes with fault-tolerance that ensure the correct execution of business processes. Although business process languages are capable of supplying fault tolerance patterns, implementation needs an infrastructure for error detection and recovery mechanisms.

To this end, the Deployment fault-tolerance infrastructure task, detailed in Chapter [5], is in charge of the supporting fault tolerance in executable business processes. The contribution consists of the definition of various fault-tolerant patterns for the implementation of executable business processes. These patterns have been equipped with a variety of recovery mechanisms based on replication (dynamic binding of services), roll-back and check-pointing, and software diversity (NV-Versioning) techniques. The contribution also provides error-detection mechanisms. The error-detection mechanisms have been enhanced with a fault diagnoser based on Model-Based Diagnosis (MDB) techniques in order to identify and isolate fault services. MDB has been developed using Constraint Programming for the automatic identification of faulty services at runtime. Subsequently, the faulty services trigger recovery mechanisms that are responsible for delivering an alternative to these services by means of the corresponding fault tolerance patterns. Therefore, this contribution constitutes a fully automated approach for the fault tolerance of executable business processes.

3.3 Related work

There are only seminal initiatives in the BPM arena that bridge the gap between BPM and risk management. In [Jakoubi and Tjoa, 2009], a notation-independent model is proposed as a reference model. Likewise, in [Sackmann, 2008], current risk management methods are extended by bridging the gap between the business process view and the more technical view of IT risks. These approaches propose theoretical reference models in an
attempt to fill the void between business and risk domains. In other respects, Neubauer et al. (Neubauer et al., 2005) propose a framework for the analysis of the security of business processes from the point of view of cost-benefit. Their framework proposal is defined for the integration into any business process management approach. In (Sienou et al., 2007), a framework is presented that unifies risk management and business process management. Their approach is limited to the presentation of various stages of the framework and how it works from a theoretical point of view. Other initiatives are focused on particular aspects of risk management, for instance risk assessment at design of business process models (Cope et al., 2010), whereas most approaches are focused on provisioning enhancements of business process languages through new domain-specific languages to aid and improve the risk assessment of business process models. The related work of each referenced contribution is introduced alongside the current contribution in the next chapters.

3.4 Summary and Discussion

This chapter presents an overview of OPBUS framework as an enhancement of BPM life-cycle. In this way, OPBUS strives towards improvement of the development of business process at design and runtime by means of risk management and fault tolerance. Its contribution is in the form of the integration of fully automated techniques for the risk assessment, and treatment of business process models at the design stage. On the other hand, executable business processes have been equipped with automatic fault diagnosis and recovery techniques for the achievement of fault tolerance at runtime.

Although certain initiatives in the literature review exist that deal with the integration of BPM and risk management, these initiatives are limited by theoretical, and non-formal model approaches with an absolute lack of provision of tools for the automation of the risk management tasks in the business processes. Furthermore, most of these initiatives are only focused on one specific task of the risk management, such as risk identification, risk analysis, or selection of controls.
Chapter 4

Automatic Diagnosis of Non-Conformance of Risks in Business Process Models

*It always seems impossible until it is done.*

Nelson Mandela.

4.1 Context and Motivation

As stated in Chapter 3, the aim of this contribution is to support the automatic diagnosis of business process models to identify non-conformance of risks according to the established objectives.

Organizations are becoming more and more security aware, although still only few assess their security risks (PricewaterhouseCoopers [2011]). Nevertheless, there exist a vast number of methodologies available, such as *MAGERIT®* (Spanish Government - Presidency Ministry, 2006), *CRAMM®* (UK Government, 2002), *COSO* (COSO, 2004), *CORAS* (Lund et al., 2011), and standards such as ISO/IEC 27000 series, NIST SP 800-30, AS/NZS 4360:2004, BS 7799-3:2006. The main aims of these methodologies include the identification of: risks that could compromise the normal work of the organization; what assets could be affected; and what remedial action(s) must be adopted. Although these methodologies deal with risks in general, in the current thesis dissertation, risks refer to IT security risks. Each methodology and standard provides its own metrics and catalogues to carry out risk management. Nevertheless, these approaches are focused on the risk management aspects and disregard any link with BPM. In addition, the implementation of the majority of these methodologies supposes a manual, informal, textual, detailed and complex process, which demands a high investment in resources for the organization.

Due to the manual, informal, and textual nature of risk management methods, it is extremely difficult to include risk management within BPM. Generally, business process
models are built considering only business goals, while in contrast, security risks are assessed and included in second place. This separation creates a gap between business process models, objectives, and risks. This gap makes it hard to ensure that business process models are built in accordance with the objectives and even to determine whether business processes are in conformance with risk analysis carried out.

Ideally, a common model would be available that would enable business and security specialists to collaborate. These models could help towards ascertaining, the conformance of security decision-making with regard to business goals. According to the standard Business Motivation Model (BMM) (Object Management Group (OMG) 2007), a business goal is defined states or conditions of the enterprise to be brought about or sustained to do. Following the BMM standard, an objective is an attainable, time-targeted, and measurable target that the enterprise seeks to meet in order to achieve its goals. Thus, a objective is quantified as a goal. For instance, a business goal related to risk could be defined whereby the risk of a business process could not exceed an acceptable risk level, while a objective could state that the risk of a business process must not exceed an established acceptable risk level of 85.

Certain approaches exist in the reviewed literature that strive to automate security and risk management within business process models. Several approaches are given in (Cope et al., 2010)(zur Muehlen and Ho, 2005)(Lambert et al., 2006)(Churliov et al., 2006), although these are focused on modelling Domain-Specific Languages (DSLs) to aid risk assessment tasks in graphical business processes. These languages are only descriptive, and they have been applied in a manual way for any BPM approach. Thus, these approaches do not provide automatic mechanisms for the risk assessment of business processes.

Our contribution pursues the quality improvement of business process management by means of the risk-aware development of business process models. To this end, we have focused on the Design and Analysis phase (cf. Figure 1.2) by providing a risk model as an extension to business process models where risk information is gathered. In addition, we provide verification techniques for the support of the automatic risk assessment of business process models and for the diagnosis of business process models to check the conformance with regard to business objectives.

Figure 4.1 describes our proposal to ensure a risk-aware business process development in the form of a business process. The figure includes actors involved in the process; activities of the business process are related to the actor responsible for its execution. In the case of activities without an actor, the activities are performed automatically. Firstly, we receive information (referred to as the Define Business Processes task in the diagram) through the business analysts about the processes (referred to as the Business Process Model data object) and the objectives (referred as Objectives) to be evaluated. In this case, business analysts are responsible for defining business process models and the objectives to be achieved. Security analysts can then introduce information into the extension whereby business process elements are quantified (referred to as the Introduce value model elements task) and information about threats, vulnerabilities and countermeasures are incorporated (referred to as the Introduce threat/vulnerabilities and Introduce countermeasures tasks). This information forms the base of the risk model (referred to as
the Risk Model data object) of the business process under assessment. At this point, an automatic risk assessment of the model is carried out. This assessment is separated into two main steps. Firstly, business processes plus the risk model are assessed to determine the value of risks for each business process. Once the risks are determined, a diagnosis (referred to as the Diagnose workflows task) is carried out to identify which potential execution flows within the business process are non-conformance to the business objectives established (for instance the acceptable risk level). A fine-grain diagnosis (referred to as the Diagnose activities/artifacts task) is then carried out where activities or artifacts in the business process model are designated as non-conformance to the business objectives and established risks.

Therefore, the main aims in this contribution can be summarized as the following:

- The definition of a risk model as an extension of business process models that enables the risk-aware design of business process models by means of integration of three perspectives: business objectives, business process model, and security risks.

- The definition and development of verification tools that enable the automatic risk assessment and the diagnosis of non-conformance of risks regarding the objectives established within business process models.

In following sections, the contributions are described in depth. In Section 4.2, the risk model extension for business processes is detailed.

4.2 Risk-Aware Business Processes

Despite the existence of a wide range of business process modelling languages: Flowcharts (Harrington et al., 1997), Petri Nets (Salimifard and Wright, 2001), Event-driven Process Chain (EPC) (van der Aalst, 1999), UML Activity Diagram (Dumas and Hofstede, 2001), Data Flow Diagram (DFD) (Kendall, 1995), Business Process Management Notation (BPMN) (Object Management Group (OMG), 2011), and IDEF3 (Bosilj-Vuksic and Hlupic, 2001), not all business process modelling languages support the same features. For instance, the BPMN and UML activity diagrams may separate the processes into
CHAPTER 4. DIAGNOSIS OF NON-CONFORMANCE OF RISKS

scenarios where the process is performed for a particular participant. However, IDEF3 and Petri Nets do not support this feature.

Currently, BPMN is the standard promoted by OMG and constitutes the most widely used and complete language for the representation of business processes on the IT market. BPMN can support all features of other languages, and even provides new features to manage roles, faults, errors, and so on. Furthermore, the BPMN specification provides extension points to enhance the language by including new artifacts and properties for the modelling elements.

In the revised literature, certain approaches, such as (Rodríguez et al., 2006) (Menzel et al., 2009) (Wolter et al., 2009), introduce extensions which specify security properties (authorization or trust) in business process modelling languages. Nevertheless, these approaches are only focused on presenting graphical extensions to enhance the languages with specific security annotations. None of these approaches takes into account the previous analysis of the business process and the conformance of the security properties with regard to the business requirements.

None of the modelling languages for business processes defines specific issues concerning risk management. Business process languages must be enhanced through the integration of other aspects such as objectives and risks. According to (Menzel, 2010) there are three approaches to providing an enhancement of a modelling language:

1. Light extension - Providing an extension of the language by means of the usage of the extension points provided by the language. In (Menzel et al., 2009), Menzel et al. provide an extension to specify security requirements in BPMN models. However, this approach is closely coupled to the technology since the security configurations provided are only valid for the SOA environment. Furthermore, not all modelling languages support extension points to enhance the modelling elements.

2. Heavy extension - Providing an extension of its meta-model. In (Rosemann and zur Muehlen, 2005), Muehlen et al. present an extension for the integration of risk into EPC models. The main disadvantage of this approach is that the extension is created only for this language and only for this meta-model.

3. Definition of a new language - Providing a new language. This modelling language provides new elements and contains specific redefined elements. In (Cope et al., 2010), the authors provide an extension of the BPMN meta-model composed of three new diagrams to support the risk assessment of BPMN models.

4.2.1 A Risk Model Extension of Business Process Models

We have utilized a BPMN-based business process meta-model, as shown in Figure 4.2. The meta-model is a sub-set of the complete set of elements of BPMN since complex elements, such as compensations handlers, links and transactions, are rarely used in business process designs (Chinosi and Trombetta, 2012). Although this meta-model has been inspired by BPMN, this proposal supports the common features of every modelling language in order to make the approach as compatible as possible with other modelling languages. The reason why a generic meta-model is used to provide an independent-modelling language
4.2. RISK-AWARE BUSINESS PROCESSES

Figure 4.2 – Business Process Meta-Model.

approach. This meta-model describes the basic entities and relations of a modelling language for the definition of business processes from a control-flow perspective.

In order to enhance the language, we define a light extension for the business process meta-model. For a better understanding, it is crucial to introduce some vocabulary concerning the risk assessment. The following concepts have been extracted from the standards ISO/IEC 73:2009: Risk management Vocabulary and ISO/IEC 27005:2001: Information technology - Security techniques - Information security risk management:

- An asset is defined as any tangible or intangible thing that adds value to an organization.
- Vulnerability is an intrinsic property of something resulting in susceptibility to a risk source that can lead to an event with a consequence.
- Threat is a potential cause of an incident that may result in harm of systems and organization.
- A risk is defined as the effects of uncertainty on the achievement of the goals.
- Risk treatment is a process of selection and implementation of controls which attempts to modify the risks.
- Control or countermeasure is something that modifies a risk.
- A risk criterion is a term of reference used to evaluate the importance of a risk.

Regarding business processes, the previous concepts can be redefined as following:
• **Assets** are the elements (activities, artifacts, pools, etc.) that provide value for the performance of business processes.

• **Vulnerabilities** can be defined as the weaknesses of a business process that could compromise correct operation of the business processes or of the information used for the business process.

• **Threats** are potential situations or actions that can affect the business process, thereby causing damage to the organization.

• **Treatments** are specific countermeasures or a set of countermeasures that organizations use to reduce the effects of a threat.

• **Countermeasures** are the appropriate measures which help to reduce the impact of an individual threat.

• **Risk** is the estimation of the degree of exposure to a threat materializing on one or more business processes and causing damage to the organization.

As stated earlier, risk assessment is the process of analysing and evaluating the risks of assets. A basic risk assessment implies at least the following tasks (the title in brackets indicates the activities in Figure 4.1):

1. **Risk identification**: Identify and value assets (Introduce value of model elements), threats and vulnerabilities in business processes. (Introduce threats and vulnerabilities).


We propose an extension that integrates business objectives (Business Motivation Model), business process model (Business Process Meta-Model), and risks (UML Profile for QoS and FT) in the same meta-model as shown in Figure 4.3. This meta-model proposes an extension for business processes according to the concepts of the UML Profile for Modelling Quality of Service (QoS) and Fault Tolerance (FT) Characteristics and Mechanisms Specification (hereinafter UML profile) (Object Management Group (OMG) 2009), and BMM (Object Management Group (OMG) 2007). The UML profile provides a set of generic concepts in order to develop risk assessment capabilities within IT systems. In other respects, BMM provides a model to define and develop business plans. Business plans are carried out in a final stage by business processes. Thus, BMM enables the identification of factors and relations to define business plans, and of how to achieve and assess these plans. The most relevant factor for our proposal is risk. Risk concern is considered a crucial impact factor in the achievement of business plans.

The extension defines three kinds of relations between models: (1) green lines that relate BMM and UML profile; and (2) red lines that relate Business Process meta-model with UML profile; (3) light-blue lines that relate BMM with Business Process meta-model.
4.2 RISK-AWARE BUSINESS PROCESSES

Figure 4.3 – Risk Model Extension.
CHAPTER 4. DIAGNOSIS OF NON-CONFORMANCE OF RISKS

<table>
<thead>
<tr>
<th>Relation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red line</td>
<td>Describes which elements of business process models (Activity and Data object) and which elements from UML profile (Threat Scenarios) are included within business process models for the risk assessment.</td>
</tr>
<tr>
<td>Green line</td>
<td>Describes the relation between objectives from BMM and the risks concepts from UML profile. The risk from UML profile can be used as an impact factor in the assessment of business plans in BMM. UML profile defines risk criterion as the criterion against risk must be evaluated in order to decide whether risk is acceptable or not. A risk criterion can be observed as a objective to achieve from BMM since we can establish an objective as a risk criterion that must not be exceed by any business process.</td>
</tr>
<tr>
<td>Light-blue line</td>
<td>Describes how Objective and Assessment from BMM are related to business process elements (Process). Objective is used to establish the statements to achieve for a business process while the assessment is used to state a impact factor, that is, a risk for a business process.</td>
</tr>
</tbody>
</table>

Table 4.1 – Relations in the extension meta-model

The extension provides the basic entities, relations, additional artifacts and properties necessary to carry out a risk assessment in business processes. The elements described for this extension have to be stated by security analysts as indicated in Figure 4.1. These elements of the extension are formalized and detailed in the sections below.

4.2.1.1 Notation and Metrics for the Estimation of Assets

A business process consists of a set of activities performed in a specific order by a participant. An activity is a unity of work that needs data (if any) in order to be performed. This data is introduced by means of data objects (artifacts in BPMN). The execution flow of activities is specified through arrows (sequence flows in BPMN) and control flows (gateways in BPMN) between the activities. For a better understanding, the formal definitions of the potential execution flow and business process model are introduced below.

**Definition 4.2.1. Potential Execution Flow (PEF).** A potential execution flow $F_i$ is a finite sequence of activities and data objects performed to achieve an objective of a business process $bp_i$. $F_i$ is defined by the tuple $(A, S)$ where $A$ is the list of activities $\{a_1, a_2 \cdots a_z\}$ performed in order by the flow. An activity is defined as a unit of work to be performed. $S$ is a set of artifacts $\{s_1, s_2 \cdots s_k\}$ used during the execution of a business process. An artifact is defined as a data object flowing through a business process.

In Section 4.3 a set of PEFs and workflow patterns (i.e. sequential, parallel, loops, etc.) are described (cf. Table 4.3 for the risk estimation of business process models.
4.2. RISK-AWARE BUSINESS PROCESSES

Figure 4.4 – Example of Business Process Model.

An example of a business process model, composed of two activities \((A_1\) and \(A_2\)), one artifact \((Data_1)\), and two PEFs \((F_1\) and \(F_2\)), is shown in Figure 4.4. We assumed that business processes are well-designed without structural faults, since structural faults are identified using the approaches in (Borrego et al., 2009; Sadiq et al., 2000; Varela-Vaca et al., 2010b).

Definition 4.2.2. PEF model. A business process model \(bp_i\) is composed of a set of PEFs \(\{f_1, f_2, \ldots, f_s\}\) within the business process. Every \(bp_i\) contains at least one PEF.

These processes are represented by pools (participants), a set of activities, data objects (artifacts), gateways (control flows) and connectors that define the structure (potential execution flow) of the business processes. When considering a well-designed business process, the activities and artifacts constitute the crucial elements its performance. From our point of view, activities and artifacts are the main assets to be assessed, since activities provide the operations to be performed, and artifacts provide the data. Therefore, our approach focuses on activities and artifacts of business processes as the principal assets under assessment.

It is first necessary to introduce certain metrics that allow the estimation of assets. The UML profile introduces the concept of *Asset value* in order to enable the estimation of an asset, such as that shown in Figure 4.3. The UML profile does not specify how to precisely estimate an asset. In general, there exist two main approaches:

- **quantitative** approach, where the asset value is defined through a range of values.
- **qualitative** approach, where the asset value is defined through a set of categories.

The main advantages of using a quantitative approach over a qualitative approach is that the quantitative scale is more accurate than a qualitative scale, and that a quantitative scale can always be transformed into a qualitative scale if required.
As a first approach, we enhance the business process meta-model to include new properties in the extension where the Asset Value is attached to the activities and artifacts of the business process. This property can be separated, for instance into three sub-metrics according to the three security dimensions established in ISO/IEC 27004 (ISO 2009a): integrity, confidentiality, and availability. This separation could provide a better estimation of risk with regard to these three typical dimensions of security, although other dimensions can be adopted, such as availability, authorization, non-repudiation. However, a global value of the asset can be obtained using a combination of the sub-metrics for instance by adding the values together.

Providing a fixed value for each dimension is unrealistic and highly complex. Hence, we have assumed a quantitative approach to these metrics by means of a range of values ranging from 1 to 10, also represented by [1-10]. Thus an activity or artifact should be estimated by means of a range of values that indicates the minimum and the maximum value allowed for each metric. In Figure 4.5, a business process is shown where the activities $A_1$ and the artifact $D_1$ are instantiated with the value of integrity, confidentiality and availability. The BPMN does not provide mechanisms to show properties of the elements in the graphical model. Therefore, in order to show these values, text annotations are linked to the activities with this information. The following definition introduces the formal definition of activities of a business process.

**Definition 4.2.3. Asset values of business process elements.** An activity or artifact $A_i$, which belongs to a business process, has an associated tuple $(E)$ where $E$ defines a set of metrics $(D_1, D_2, \cdots, D_n)$ for the estimation concerning security dimensions of the activity or artifact.

The implementation is defined by the triple $(I, C, A)$, which provides an estimation of the activity or artifact with regard to the security properties of integrity ($I$), confidentiality ($C$) and availability ($A$). We have proposed the separation of the asset value into three
4.2. RISK-AWARE BUSINESS PROCESSES

dimensions, depending on the type of threat. Thus, a threat could affect an asset in one, two or three security dimensions, although our proposal could be extended in order to support other aforementioned dimensions. Hence, the risk could be estimated by means of the direct application of the threat to the asset dimension. This may prove highly useful in the risk treatment stage since we know from the risk assessment which security dimensions must be treated, and hence adequate countermeasures could be selected for the affected dimensions.

The risk assessment strives to evaluate the extent to which activities and artifacts can be affected by threats and vulnerabilities, by determining how many threats can compromise the correct operation of a business process. To this end, we introduce an extension that describes information about threats and metrics that enables the measurement of threats and vulnerabilities in business process models.

4.2.1.2 Notation and Metrics for the Annotation of Threats and Vulnerabilities

The UML profile introduces the concept of scenario attached to the assets, such as shown in Figure 4.3. These scenarios identify unwanted incidents which are concerned with exploring the threats and vulnerabilities of the asset under assessment. In our approach, the scenarios provide and gather information about threats that compromise the correct operation of the elements of the business process.

We have enhanced the business process models with a new artifact. This artifact, called Threat Scenario, allows the specification of threats, vulnerabilities and treatments attached to the business process, as shown in Figure 4.6. This example shows a business process with a threat scenario composed of two sets; the first set is utilized to gather information about threats, and the second set is employed to specify countermeasures. In our approach a set of vulnerabilities can be associated to each activity or artifact independently. Figure 4.6 shows two activities, $A_1$ and $A_2$, where $A_1$ is affected by vulnerabilities $V_1$ and $V_2$, and activity $A_2$ is affected by vulnerabilities $V_4$ and $V_5$. It is assumed that business process models under assessment have at least one threat scenario attached. In the same way, a threat scenario can be utilized by several business processes.

The main advantage of introducing scenarios is to group all the information related to the threats and treatments (such as those proposed by the UML profile) on the same artifact independently of business processes. Furthermore, this artifact prevents the repetition of evaluations. For instance, for a business process composed of 20 activities, 10 artifacts, and 3 threats; without threat scenarios, the threat metrics should be stated by each activity and artifact independently, which supposes $20 \times 3 + 10 \times 3 = 90$ statements. An example of a business process without any threat scenario is shown in Figure 4.7. This example shows how the metrics related to the threats have to be attached for each activity separately. By contrast, in the case of using threat scenarios, the number of statements is $20 + 10 = 30$. This problem worsens when treatments are considered since these have to be stated and related to the threats introduced in the model. For instance, if we take into account 5 treatments, one for each threat, then the problem increases to $20 \times 5 \times 5 + 10 \times 5 \times 5 = 750$ statements. However, the number of statements concerning threat scenarios is $20 + 10 + 5 = 35$. Therefore, the utilization of threat scenarios in-
CHAPTER 4. DIAGNOSIS OF NON-CONFORMANCE OF RISKS

Figure 4.6 – Threat Scenario attached to a business process model.

... introduces a crucial improvement in the number of statements of the threat, vulnerabilities and treatments.

Definition 4.2.4. Threat Scenario. A threat scenario $TS_p$ is defined by a tuple $(P, R, T)$ where $P$ is the set of business processes associated to this scenario; $TH$ is the set of threats $\{th_1, th_2, th_3, \ldots, th_n\}$ of this scenario; $T$ is a set of treatments $\{t_1, t_2, t_3, \ldots, t_n\}$ related to the threats.

$$TS_p \subseteq T \times R$$
$$\forall t_i \in TS_p (\exists th_j \in TH | (t_i, th_j) \in TS_p)$$

As mentioned above, certain metrics that allow the assessment of the risks of materialization of these threats need to be included. Each methodology or standard proposes its metrics for the estimation of threats. For example, FMEA (Papadopoulos et al., 2001) uses three metrics: severity, occurrence and detection. These metrics range on a scale from 1 to 10. In other respects, MAGERIT® (Spanish Government - Presidency Ministry, 2006) uses only two metrics: impact and frequency without specific scales. NIST in (Gary Stoneburner and Feringa, 2002) proposes two groups of metrics to describe the exploitability and the impact using a qualitative scale. The UML profile defines a threat based on certain metrics which refer to frequency and consequence. We have adopted an approach based on the metrics frequency and consequence as defined by the UML profile. An example of these metrics is shown in Figure 4.6 where the threat $R_1$ has a frequency of between 2 and 4 and a consequence of between 4 and 5. The approach can be adapted to other metrics by extending the properties of the threats in the model. For a better understanding a formal definition of a threat and of threats associated to an element of business processes is given below.

Definition 4.2.5. Threat. A threat $TH$ is defined by a tuple $(ID, M, D, V)$, where $ID$ indicates a unambiguous name for the threat; $M$ is a set of metrics $\{M_1, M_2 \ldots, M_n\}$
4.2. RISK-AWARE BUSINESS PROCESSES

Figure 4.7 – Evaluation threats without threat scenarios.
in relation with the threat; and \( D \) represents the security dimensions \( \{D_1, D_2, \ldots, D_j\} \) that could be compromised; a threat \( th_i \) has associated one or various vulnerabilities \( \{V_1, V_2, \ldots, V_j\} \) where a vulnerability is a flaw associated to a threat \( th_i \) that could compromise the correct operation of a business process;

\[
TH \subseteq R \times V
\]

\[
\forall th_i \in TH (\exists v_j \in V | (th_i, v_j) \in TH)
\]

In general, the estimation of these metrics relies on the expertise of various business and security stakeholders and their experience to accurately estimate this data (Fenz et al., 2009; Neubauer and Heurix, 2008). Furthermore, it is crucial to take into consideration the dependence between these metrics as fundamental risk factors in the determination of risk. Techniques for the evaluation of the dependency of risk factors could be used, such as AHP (Fang and Marle, 2012), ANP (Khademi et al., 2012), DEMATEL (Wen-Hsien et al., 2012), and Risk-Rank (Alpcan and Bambos, 2009). We assume that the risk formulae and the estimated values of risk factors already reflect this dependency.

Our proposal is framed within a quality-improvement life cycle. In a first iteration, an estimation of metrics based on the expertise of business analysts and security experts is assumed. However, these estimations could be more accurate if data on previous assets were taken into consideration. In subsequent iterations, we propose using monitoring capabilities in order to catch information related to these metrics during the run-time phase of business processes. We also propose extending the proposal to use databases and ontologies; such as in (Menzel, 2010), with information of logs and tracking of assets. In this paper, we assume that metrics information is provided by an estimation by business analysts and security experts in order to carry out a first estimation of risk or based on the data obtained in previous iterations.

In other respects, vulnerabilities are seen as a weakness related to certain threats that may compromise the correct operation of an asset within a business process. The UML profile considers these vulnerabilities as weaknesses that can be exploited by one or more threats. Following the UML profile, we have enhanced the specification of the business process model by including a list of vulnerabilities attached to the threats, as shown in Figure 4.6, where threats \( R_1 \) and \( R_2 \) are attached to the vulnerabilities \( V_1 \) and \( V_2 \) respectively. In the risk meta-model, a relation between threats, vulnerabilities and assets is defined. Activities and artifacts of business processes could be associated to a set of vulnerabilities. These vulnerabilities can be related to a set of threats into a threat scenario. However, for our purpose, vulnerabilities merit only a description since they do not give extra information on the estimation of risk, although, certain available catalogues, such as NIST vulnerability database (National Institute of Standards and Technology, 2011) and ISO/IEC 27005 - Annex C (ISO, 2009b), can be adopted to extract more information about the vulnerabilities to be included in our approach. However, these catalogues also describe a vast quantity of technical information unnecessary for the assessment of the business process.

**Definition 4.2.6. Vulnerabilities associated to business process elements.** An activity or artifact \( A_i \) which belongs to a business process is associated to a tuple \( (E, V) \)
where $E$ defines a set of metrics which contain the estimations previously defined, and where $V$ defines the set of vulnerabilities $\{t_1, t_2 \cdots t_n\}$ that affect the activity or artifact.

In real cases, it is possible to find the same threat associated to several vulnerabilities, and hence various threats can be introduced in the threat scenarios associated to a particular vulnerability with specific values of frequency and consequence. For instance, threats of management error could be related to a vulnerability of work overload or a vulnerability of lack of training. In this case, the threat could be evaluated using different values for the metrics, such as:

- Threat: Management Error $\rightarrow$ Vulnerability: Overload  
  - Frequency: $[1-3]$  
  - Consequence: $[2-3]$

- Threat: Management Error $\rightarrow$ Vulnerability: Lack of training  
  - Frequency: $[3-5]$  
  - Consequence: $[3-5]$

This example is supported by our approach. However, if there are two activities affected by the same threat but with a different frequency or consequence, then this situation could present a limitation in our approach. In order to consider these situations, threat scenarios should be avoided and threats should be configured individually for activities and artifacts. As mentioned earlier, this could greatly increase complexity. Due to modelling considerations, we prefer to avoid this additional complexity by means of using threat scenarios where this situation can be simulated by two different threats with different values of frequency and consequence.

In security, threats and vulnerabilities are counteracted by treatment plans. These mechanisms introduce a course of action that modifies the achievement of a threat. Countermeasures already in place should be taken into consideration in the estimation of risk. The following section shows how countermeasures and treatments have been included in the extension.

### 4.2.1.3 Notation and Metrics for the Annotation of Treatments

In a previous section, the concepts of threat scenarios have been defined. These scenarios are composed of two sections: the first to specify the threats, and the second to specify the treatments. In this particular case, we have enhanced the threat scenarios to support the specification of countermeasures. Figure 4.8 shows a threat scenario with a treatment composed of $T_1$ and $T_2$ countermeasures.

A countermeasure is defined as a specific mechanism to modify a risk. However, no technique currently exists which indicates that a threat is treated by one specific countermeasure, and it is necessary therefore to include certain elements in the model that indicate which threat is being treated by which countermeasure. To this end, we have introduced a new element which permits threats and countermeasures to be connected
in the same threat scenario. In Figure 4.8 there is an example which indicates that the
countermeasure $T_1$ treats the threats $R_1$ and $R_3$, and the countermeasure $T_2$ treats the
threat $R_5$.

Generally, the first risk estimation in the risk assessment is made without consid-
ering countermeasures. Nevertheless, there exists the possibility that the organization
may have countermeasures already deployed. In our approach, these countermeasures are
considered for risk assessment. The estimation of risks must take countermeasures into
consideration since a countermeasure already in place can reduce the frequency of materi-
alization and reduce the consequence of one threat. Countermeasures need to define these
metrics that indicate by how much the effects of the threats are reduced (in frequency
and consequence).

The UML profile defines the concept of Risk Reduction as how the value can reduce the
risk. In our approach following the UML profile, countermeasures have been equipped
with a metric called Risk Reduction which indicates how that reduction influences the
frequency and the consequence of a threat. Risk reduction could modify the frequency
and consequence independently; however we consider that this reduction is applied in the
same percentage for both properties of threats. This metric has a domain range from 1 to
100 that indicates the percentage of reduction. Moreover, we have assumed that the risk
reduction affects the security dimensions stated in the threat to be treated. In addition,
we have included a new metric in the countermeasures to take into account the cost of
implementation. A formal definition of countermeasure is given below.

**Definition 4.2.7. Countermeasure.** A countermeasure $K_i$ is a mechanism that mod-
ifies the effects (consequences) of a threat or reduces the frequency of materialization.
A countermeasure is defined as a tuple $(TT, RR, C)$ where $TT$ is a set of threats
${\{th_1, th_2 \cdots th_n\}}$ associated to the treatment. $RR$ is the risk reduction (as percentage),
and $C$ is the countermeasure cost in monetary units.
In the same way as mentioned in Section 4.2.1.2, the estimation of these metrics relies on the expertise of a number of stakeholders involved in the assessment processes.

The extension has been introduced as a common view of the business model, business process, and risk management. Business processes are built taking into consideration business goals. These goals may refer to aspects of risks. To this end, information about goals has to be aligned with the business processes in order to express what the business and security analyst needs and what the business process reflects. In the following section, we introduce how goals and other metrics have been considered in the extension.

### 4.2.1.4 Notation and Metrics for the Establishment of Objectives

The integration of goals and cost information into the business process models has already been proposed in Korherr and List (2007). However, this information is merely descriptive and is limited to cost and time aspects. In our approach, these goals also refer to security issues. The UML profile does not provide mechanisms to gather information on goals. Therefore, we propose an enhancement of business processes models by including the concept of objective, as defined by the Object Management Group (OMG) (2007) as a mechanism to express the constraints that the customer needs with respect to the business process.

**Definition 4.2.8. Objective.** A objective establishes a requirement that the business should ensure.

In our approach, the extension is equipped with objectives that are attached to the business process. The objectives can be specified by logical constraints using a basic grammar, such as that defined by the following grammar in BNF. An example of objectives can be seen in Figure 4.9, where Objective establishes the **Acceptable Risk** of BPi at 200 and also indicates that the risk of BPi must be lower than the **Acceptable Risk**, and that the maximum cost (BPi.TotalCost) for the business process cannot exceed 2,000 monetary units, which includes both fixed costs and additional costs after introducing countermeasures.

\[
Constraint ::= Expr~OP~Expr \hspace{1cm} (4.5)
\]
\[
\quad \mid Expr~\neg Expr \hspace{1cm} (4.6)
\]
\[
OP ::= BoolOp~\mid~RelationOp~\mid~MathOp \hspace{1cm} (4.7)
\]
\[
BoolOp ::= \lor~\mid~\land \hspace{1cm} (4.8)
\]
\[
Expr ::= Constraint~\mid~val \hspace{1cm} (4.9)
\]
\[
RelationOp ::= <~\mid~\leq~\mid~=~\mid~>~\mid~\geq~\mid\neq \hspace{1cm} (4.10)
\]
\[
MathOp ::= +~\mid~\neg~\mid~\times~\mid~\div \hspace{1cm} (4.11)
\]

The risk estimation is determined through the combination of the risk obtained for each activity, each artifact, and the control-flow perspective of the business process. The estimation of risks produces a value that indicates how much an element in the business
is affected by specific vulnerabilities. The risk estimation only produces a value which has to be compared with the acceptable risk level. This risk threshold indicates whether the value from a risk value is acceptable or not. In general, this risk threshold is fixed globally for the risk assessment and is used in the comparison of acceptable risk level with the risk of business processes. It is therefore crucial to introduce any metrics that enable a comparison of risks to be carried out. The UML profile introduces the risk criterion. Risk criterion is defined as the risk objective to achieve for the risk assessment, and is crucial to decision-making in risk assessment.

The extension has been equipped with a metric (Risk Criterion) as part of the objectives to specify the acceptable risk levels for business processes. In our approach, this metric is quantified as a maximum numerical value that indicates the maximum allowed value. In the case when the risk of a business process exceeds this risk threshold, then the process is in non-conformance to the objectives of risk thresholds. Security analysts have to decide on the course of action: transfer, avoid, mitigate, or accept the risk evaluated. In Figure 4.9 a business process is given two hundred as the Acceptable Risk value which indicates the risk threshold permitted.

In this section a risk model of the business process models has been introduced. This risk model presents an extension together with a graphical notation for business processes in order to provide information relative to risks and objectives. The aim of this metamodle is to provide sufficient information to automate the risk assessment of business processes. To conclude this section, Table 4.2 shows a summary of the new modelling elements introduced by the extension. The table indicates the name of the element, a description, the set of metrics related to the element, the scale utilized to measure the metrics, and a graphical representation.
### 4.2. RISK-AWARE BUSINESS PROCESSES

<table>
<thead>
<tr>
<th>Name</th>
<th>Metric</th>
<th>Scale</th>
<th>Description</th>
<th>Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asset Value</td>
<td>I, A, C</td>
<td>[1-10]</td>
<td>Permits the estimation of an activity or artifact for various security dimensions.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>V</td>
<td></td>
<td>List of vulnerabilities that affect the activity or artifact.</td>
<td></td>
</tr>
</tbody>
</table>

| Threat Scenario | ~      | ~      | Permits the inclusion of information about threats and treatments.         |          |

<table>
<thead>
<tr>
<th>Threat</th>
<th>Frequency</th>
<th>Consequence</th>
<th>Permits the estimation in frequency and consequence of threats that affect the business process.</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td></td>
<td></td>
<td>Describes the security dimensions that a threat could affect.</td>
<td></td>
</tr>
<tr>
<td>V</td>
<td></td>
<td></td>
<td>Describes the vulnerabilities related to the threat.</td>
<td></td>
</tr>
</tbody>
</table>

| Treatment       | Risk Reduction | [1-10] | Indicates the risk reduction of a countermeasure that is introduced to treat the threats. |          |
|                 | Cost          | Monetary units | Indicates the cost of the countermeasure.                                                   |          |

| Objectives      | Objective | String | Indicates the requirements of the business process.                                       |          |
|                 | Acceptable Risk | Integer | Indicates the value of allowed risk.                                                      |          |

Table 4.2 – Summary of the elements of extension for business processes.
4.3 Diagnosis of Non-Conformance of Risks

In the previous section, we introduced an extension that provides an enhancement of the business process modelling language that incorporates the risk annotation of the business processes. The main idea for the automation is to provide a mechanism that, with minimal human intervention, determines the risks (risk assessment) and identifies which elements are in non-conformance to the acceptable risk level specified (check conformance). To this end, this section introduces the risk estimation and evaluation process, the formalization of the problem in order to identify non-conformances in business processes, and the way in which this process is automated through model-based diagnosis techniques. The diagnosis described in this section is completely transparent since we have developed a set of tools \cite{Varela-Vaca2012} that carry out this process in automatic way.

4.3.1 Risk Assessment of Business Process Models

A risk is the quantification of exposure to the materialization of a threat for one or more activities or artifacts in business processes. The risk assessment is based on the estimation of the qualification or quantification of risks. The risk estimation is carried out by means of a formula that enables a risk value to be obtained. This value gives an idea about how much an asset under assessment is affected by specific threats. The risk of an asset has to be calculated against every threat. There exist numerous risk assessment methods that are widely used in the estimation of risks. Each method provides or adopts its own risk formula. In general, the risk formula is defined on metrics related to an asset and the threats. For example, FMEA uses the concept of Risk Priority Number (RPN) where the metrics used in the definition of the RPN calculation are severity, occurrence, and detection:

\[
RPN = \text{Severity} \times \text{Occurrence} \times \text{Detection}
\]  

(4.12)

The main drawback in FMEA is that RPN fails to take the value of the assets into consideration. By contrast, MAGERIT® is flexible, and, for example, it can use a risk formula that relates the asset value (V) and the threat impact and frequency:

\[
\text{Risk} = V \times \text{Impact} \times \text{Frequency}
\]  

(4.13)

In general, the risk estimation is based on a risk formula that involves certain metrics, which refer to assets and threats. Formally, the risk estimation can be defined as follows:

**Definition 4.3.1. Risk.** A risk is defined by a tuple \((\text{Formula, Metrics, Scale})\) where \(\text{Formula}\) is the formula \(f_i\) used in the estimation of risks; \(\text{Metrics}\) is a set of metrics \((M_1 \times M_2 \times M_3 \cdots \times M_n)\) involved in the definition of \(f_i\); and \(\text{Scale}\) is the range \(R_i\) of the risk formula.

\[
f_i : M_1 \times M_2 \times M_3 \cdots \times M_n \rightarrow R_i
\]  

(4.14)
4.3. DIAGNOSIS OF NON-CONFORMANCE OF RISKS

For instance, the risks can be defined for FMEA and MAGERIT as follows:

\[
FMEA = Severity \times Occurrence \times Detection \tag{4.15}
\]

\[
MAGERIT = Value \times Impact \times Frequency \tag{4.16}
\]

In our approach, a risk formula and metrics very similar to those used in MAGERIT© are adopted. Nevertheless, other risk formulae can be included in our approach. This formula is defined as indicated in the following formulae, where Dimension (D) refers to security dimensions such as Integrity, Confidentiality and Availability of the asset defined in Definition [4.2.3] and Consequence and Frequency refer to properties of a particular threat (as defined in Definition [4.2.6]).

\[
Risk_{OPBUS^D_{Asset}} = Value^D_{Asset} \times Consequence^D \times Frequency^D \tag{4.17}
\]

This formula enables risk to be estimated by considering threats that affect the specific dimensions of an asset. In the case of threats that affect various dimensions, the risk has to be estimated within the respective dimensions. This formula is versatile and extensible since it could be adapted to incorporate new parameters, for example, to consider different weights for the security dimensions. Our approach is defined as a risk-methodology-independent approach since the framework can be adapted to various risk formulae at the same time. The risk formula is selected before the automatic assessment is carried out. This process is given in detail below.

Previous risk formula has been defined without taking countermeasures into consideration. However, a risk formula may consider countermeasures. In this case the risk is called residual risk. The risk formula which considers countermeasures includes adjustments where the value of risk reduction (RR) introduced by countermeasures is subtracted from that of consequence and/or frequency of threats. Hereinafter, the formula in listing [4.18] is adopted as a proof of concept, although other formulae could be used.

\[
Risk_{OPBUS^D_{Asset}} = Value^D_{Asset} \times (Consequence^D - RR^D) \times (Frequency^D - RR^D) \tag{4.18}
\]

In our approach, risks are estimated for each activity and artifact in the business process since each element has independently associated threats. Each activity or artifact has a risk value that must be taken into consideration to estimate the risk of the entire business process. The main problem therefore becomes how to determine the risk of a business process. In (Xue et al., 2012)(Fenz et al., 2009), the authors provide similar approaches that take into consideration the topology (control-flow) of the business processes in order to determine the risk of data errors and the level of importance of resources, respectively. Our approach considers that the risk of a business process should be estimated through the combination of the risks of each individual activity or artifact regarding the control-flow of the business process model.
A business process has been defined as a set of PEFs (as defined in Definition 4.2.1) and these potential execution flows define a path or sequence of execution. For example, a PEF in the business process can have less risk than other business processes due to the number of activities or artifacts that are executed in this potential execution flow. Therefore, we propose taking these potential execution flows into consideration to estimate the risks of entire business processes. In (Varela-Vaca et al., 2011c), a set of structures is proposed for the risk estimation of a business process as described in Table 4.3. These risk formulae take into account the fact that the structure of the business process is an adaptation similar to the estimation of the time-efficiency proposed in (Huang et al., 2008) and the estimation of security proposed in (Zo et al., 2010). Our implementation supports these workflow patterns, although it can be set up to other similar workflow patterns, such as that is shown in Section 4.3.3. Business processes with a complex structure can be calculated using a combination of these patterns.

The risk estimation of a business process is necessary for the identification of which PEFs and activities are in non-conformance with regard to the acceptable risk level established in the objectives. The risk estimation and the identification of non-conformances are carried out in a diagnosis phase, such as that shown in Figure 4.1.

As an illustration, a small example of one risk estimation of the business process BP1 is set out in Figure 4.10. The risks of the activities and artifacts of BP1 are assumed as already calculated for the dimension of integrity: Risk$_{A_1}$, Risk$_{A_2}$, Risk$_{A_3}$, Risk$_{A_4}$, Risk$_{A_5}$, Risk$_{A_6}$, Risk$_{A_7}$, Risk$_{A_8}$, Risk$_{A_9}$, Risk$_{A_{10}}$, and Risk$_{D_1}$. Similarly, risk can be calculated for the remaining security dimensions.

The Risk$_{BP_1}$ is dynamically built in accordance with the control-flow and the patterns defined. Firstly, the business process begins with a start event $S_1$, and is followed by a sequence of activities that start with $A_1$. The sequential pattern introduced therefore indicates that the risk is the sum of the activities that compose this sequence, divided by the total number of activities. In this case, the sequence is composed of $A_1$, $A_2$, a Parallel pattern (hereinafter $P_1$), $A_5$, $A_6$, an Exclusive pattern (hereinafter $E_1$), and finally $A_{10}$. Hence, Risk$_{BP_1}$ is given as follows:
<table>
<thead>
<tr>
<th>Pattern</th>
<th>Formula</th>
<th>Business Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequential</td>
<td>$Risk_{BP_i}^D = Risk_{F_i}^D = \sum_{i=1..n}(Risk_{A_i}^D)/n$</td>
<td>![Sequential Diagram]</td>
</tr>
<tr>
<td>Parallel</td>
<td>$Risk_{BP_i}^D = \sum_{i=1..n}(Risk_{P_i}^D)/n$</td>
<td>![Parallel Diagram]</td>
</tr>
<tr>
<td>Exclusive</td>
<td>$Risk_{BP_i}^D = MAX(\forall_{i=1..n}(Risk_{P_i}^D))$</td>
<td>![Exclusive Diagram]</td>
</tr>
<tr>
<td>Loops</td>
<td>$Risk_{BP_i}^D = MAX(\forall_{i=1..n}(Risk_{F_i}^D)$</td>
<td>![Loops Diagram]</td>
</tr>
</tbody>
</table>

$Risk_{BP_i}^F = \frac{Risk_{A_1}^D + Risk_{A_2}^D}{2}$

$Risk_{BP_i}^F = \frac{Risk_{F_1}^D + Risk_{F_2}^D}{2} = \frac{Risk_{A_1}^D + Risk_{A_2}^D}{2}$

$Risk_{BP_i}^F = MAX(Risk_{F_1}^D, Risk_{F_2}^D) = MAX(Risk_{A_1}^D, Risk_{A_2}^D)$

$Risk_{BP_i}^F = MAX\left(\frac{Risk_{A_1}^D + Risk_{A_2}^D}{2}, \frac{Risk_{A_1}^D + Risk_{A_2}^D + Risk_{A_3}^D}{2}\right)$

Table 4.3 – Basic patterns for risk determination in business processes.
CHAPTER 4. DIAGNOSIS OF NON-CONFORMANCE OF RISKS

\[ Risk_{BP1}^I = \frac{Risk_{A1}^I + Risk_{A2}^I + Risk_{P1}^I + Risk_{A5}^I + Risk_{A6}^I + Risk_{E1}^I + Risk_{A10}^I}{7} \]  

(4.19)

The main purpose of this formula is to produce a risk average for the business process which is necessary for comparison with the established acceptable risk. For instance, in the case where the risk of any activity ranges from 1 to 100, the acceptable risk is 100. By taking into consideration only the sum of all the activities, this gives rise to \(10 \leq Risk_{BP1}^I \leq 1000\). Nevertheless, this risk is over-estimated in comparison with the established acceptable risk level. Therefore, the inclusion of the average by means of the total risk of activities gives rise to \(1 \leq Risk_{BP1}^I \leq 100\). The next step is to estimate the risk of \(Risk_{P3}^I\) and \(Risk_{E1}^I\). Since it has been assumed that the business process is well-designed without structural faults, gateways are presumed to be opened and closed correctly. The risk of a Parallel pattern is estimated as the sum of the risk activities and artifacts of which each branch of the pattern is composed, divided by the total of activities of the parallel pattern. In the example there are two branches: the first branch composed of activity \(A_3\), and the second activity composed of activity \(A_4\). Therefore, \(Risk_{P1}^I\) is estimated by the following formula:

\[ Risk_{P1}^I = \frac{Risk_{A3}^I + Risk_{A4}^I}{2} \]  

(4.20)

On the other hand, the risk of an Exclusive pattern is estimated by the maximum risk between the branches. In this case, there are two branches: the first composed of activities \(A_7\) and \(A_8\), and the second branch composed of activity \(A_9\). Therefore, \(Risk_{E1}^I\) is determined as follows:

\[ Risk_{E1}^I = \text{MAX} \left( \frac{Risk_{A7}^I + Risk_{A8}^I + Risk_{D1}^I}{3}, \frac{Risk_{A9}^I + Risk_{D2}^I}{2} \right) \]  

(4.21)

Finally, by combining the three formulae, the risk becomes:

\[ Risk_{BP1}^I = \frac{Risk_{A1}^I + Risk_{A2}^I + \left( \frac{Risk_{A5}^I + Risk_{A6}^I}{2} \right) + Risk_{A5}^I + Risk_{A6}^I}{7} + \text{MAX} \left( \frac{Risk_{A7}^I + Risk_{A8}^I + Risk_{D1}^I}{4}, \frac{Risk_{A9}^I + Risk_{D2}^I}{2} \right) + Risk_{A10}^I \]  

(4.22)

The formulae are described in order to provide one way of estimating the risk of business process models. These formulae describe the risk estimation by means of PEFs within a business process model. This separation between potential execution flows is suitable for the diagnosis. In general, the diagnosis enables the identification of the parts
which fail, and determines why a correctly designed system fails to work as expected. The
diagnosis is utilized to identify which elements in the model are in non-conformance to the
acceptable risk level. In our approach, we have proposed two diagnosis steps as defined in
Figure 4.1 Firstly, the diagnosis strives to identify which PEFs are in non-conformance.
Secondly, the diagnosis strives to identify which activities and artifacts of the potential
execution flow are in non-conformance. Therefore, the diagnosis can be described as the
following steps:

1. In order to determine risks of business processes when there exist PEFs that are in
   non-conformance, a model-based diagnosis is applied to identify the activities and
   artifacts of these potential execution flows.

2. Diagnosis has to be applied to every business process that could be interconnected
to another. In any non-conformance, a diagnosis is utilized to identify precisely
where the non-conformance lies within the business process.

Finally, regarding the metrics, in our proposal, most metrics are specified by a range of
values. These ranges make our proposal more accurate with respect to those conventional
methodologies that only consider single values for the metrics, since the inclusion of ranges
introduces an a refinement into the risk estimation.

In the following subsection, a formalization of the problem of diagnosis is given. The
main purpose of formalization of this problem is to achieve the automation of diagnosis.

4.3.2 Model-based Diagnosis

As introduced in the background section, model-based diagnosis aims to identify the
reason for unexpected behaviour, or in other words, to identify the part or parts which
fail in a business process. Model-based diagnosis uses a model that describes the system,
and a observational model that provides information about inputs and outputs of the
system. In our approach, an observational model is not available since the business
processes are in design phase. Therefore, model-based diagnosis is used to check the
consistency of the business process models and risk model respect to objectives. In general,
model-based diagnosis techniques have been applied in the literature in order to identify
structural faults and activities which are unexecutable in business processes (Borrego
et al., 2009) (Borrego et al., 2010b) (Varela-Vaca et al., 2010b). In general, this problem
can be formalized as follows.

In context, our problem is a set of business processes of a BPMS that need to be
assessed. In a BPMS, a set of n business process models \{BP_1, BP_2, \ldots, BP_n\} could be
executed for all \(BP_i \in BPMS\), whereby there exists an estimated risk in the design time
t such that \(R(t)_{BP_i} = \{R(t_1)_{BP_i}, R(t_2)_{BP_i}, \ldots, R(t_m)_{BP_i}\}, 1 \leq t \leq m\).

In order to estimate a risk, an assessment is performed which determines whether
risks exceed the threshold imposed by the organization. In the following definitions, it is
assumed that risks are determined at a fixed point of time \(t_i\). Therefore, at design time
\(t_i\) for a BPMS, the associated risk is represented by \(R(t_i)_{BPMS} = R_{BPMS}\). The main
problem related to the risk assessment of a BPMS is the diagnosis of non-conformance
of risks within business processes of a BPMS regarding the objective of acceptable risk level. For a better understanding, these problems have been formalized similar to the formalization used by Cordier et al. (Cordier et al., 2004).

In a BPMS, a set of \( n \) business process models could be executed. A business process model, \( BP_k \), is composed of a set of activities that are attributed with security risks. A risk of one activity, activity \( R_{A_j} \), is represented by an \( n \)-tuple defined as the combination of a set of dimensions, such as integrity, confidentiality, availability, authorization, and authentication. In \( 4.23 \) vector notation represents the different security dimensions of \( R_{A_j} \).

\[
R_{A_j} = (R_{A_j}^i, R_{A_j}^c, R_{A_j}^a, R_{A_j}^{az}, R_{A_j}^{at})
\]

\[
(4.23)
\]

In the same way, the risk of a business process \( BP_k \) might be defined as the combination of \( R_{BP_k} \), as proposed in (Pérez et al., 2006).

A set of business goals (\( BG \)) establishes a set of constraints that relate the risks of a \( BP_i \) and its business process models. In general, organizations establish a set of business goals related to the appraised, i.e. a risk of a business process \( BP_k \) is acceptable if \( R_i^A \leq n \), where \( n \) may be a natural number.

**Definition 4.3.2. Non-conformance-of-Risk-Problem.** A non-conformance of a risk problem (NCRP) is a tuple \( (R_{BP_i}, BG, TS) \), where \( R_{BP_i} \) represents the system model to be diagnosed (for model-based diagnosis theory (de Kleer et al., 1992b), components and system model), and \( BG \) represents the business goals; \( TS \) is a set of threat scenarios \( \{ts_1, ts_2, \cdots, ts_n\} \).

The system \( R_{BP} \) is therefore unsatisfiable with regard to the \( BG \) constraints established. The diagnosis strives towards the identification of inconsistencies in the risk of an activity or artifact \( A_j \), and specifically the set of dimensions (e.g. integrity, confidentiality, availability, authorization, and authentication) within the activities that produce that inconsistency. This identification might be solved by means of fault diagnosis theory. Following the theory of consistency-based diagnosis proposed in (de Kleer et al., 1992b), this problem has been formalized as the fault diagnosis of a risk-management problem (cf. Definition 4.3.2) in the following definitions.

**Definition 4.3.3. Fault Diagnosis of an NCRP.** A fault diagnosis of an NCRP is a set of risks of activities and artifacts \( \Delta \subseteq R_{BP} \) such that \( \Delta = \{\Delta_{A_j} \cup \Delta_{A_i} \cup \cdots \cup \Delta_{A_m}\} \), where \( \Delta \subseteq R_{A_x} \) and \( x \in \{j, \cdots, m\} \).

\[
(R_{BP_i} - \Delta) \cup BG \vdash \top
\]

\[
(4.24)
\]

In our approach, the diagnosis is applied in order to check the conformance of risks of a business process models with regard to the risk objectives. The diagnosis therefore strives to determine why a business process is unexpectedly in non-conformance to the acceptable risk level. The diagnosis is separated into two steps as outlined in Section 4.1. Firstly, the diagnosis strives to identify which potential execution flows exceed the risk threshold.
4.3. DIAGNOSIS OF NON-CONFORMANCE OF RISKS

Secondly, a diagnosis is applied to the potential execution flows marked as being in non-conformance, in order to identify which activities or artifacts are in non-conformance to the acceptable risk level.

The diagnosis consists of: (1) determination of which PEFs are in non-conformance of risks with regard to the acceptable risk level; (2) identification of which activities and artifacts lie within the PEF. Following the example of Figure 4.10, the intersection between the PEFs $F_1$ and $F_2$ yields the exclusive gateway. This gateway introduces two possible execution branches: (1) $A_7, A_8, D_1$; and (2) $A_9, D_2$. $F_1$ takes branch (1) into consideration while $F_2$ takes branch (2) into consideration. In the hypothesis whereby the activity responsible for the non-conformance is $A_9$, the PEFs involved in the non-conformance have to be diagnosed. To this end, the risk of each PEF must first be estimated in a similarly way to that explained in the previous section. These risks are compared to the established risk threshold (Acceptable risk). The PEF $F_2$ is identified as responsible for the non-conformance. Subsequently, the diagnosis must be focused on the identification of the specific activities within $F_2$ that are in non-conformance to the risk threshold established. Finally, the risk of each activity within $F_2$ has to be compared individually against the risk threshold established for this business process. In the case where the activity exceeds the threshold, it is marked as being in non-conformance. When various non-conformance risk PEFs lie in the same business processes, then all the PEFs involved have to be observed.

In the case when the objective is related to the risk, there is only one possibility that the business process risk exceeds the acceptable risk level. Thus, the risk of a potential execution flow exceeds the threshold, and the set of treatments applied in the risk estimation are also insufficient to reduce the risks.

In Figure 4.11 there is an example of two business processes, $BP_1$ and $BP_2$, with two activities, $A_1$ and $A_2$, which are in non-conformance to the acceptable risk level. These activities belong to the potential execution flow $F_1$.

In order to automate the diagnosis, Constraint Programming (CP) techniques are proposed. In the following section, the diagnosis using constraint programming is given in detail.

4.3.3 Automatic Diagnosis using Constraint Programming

Constraint programming is an Artificial Intelligence technique widely used to solve diagnosis problems in various issues (Ceballos et al., 2003) (Ceballos et al., 2006) (Borrego, 2012). Constraint programming is based on the algorithmic resolution of Constraint Satisfaction Problems (CSP) as defined in Definition 2.4.1. Constraint solvers strives to propagate a value for each variable in order to satisfy the maximum number of constraints within CSP.

In our approach, the variables and domains of the problem correspond to the metrics defined by the extension proposed in Table 4.2. Constraints correspond to the objectives in order to obtain a risk of the business processes lower than the acceptable risk level. Risks of the activities and artifacts have to be previously estimated since the risk of the entire business process is the combination of these risks by means of the formulae provided
Figure 4.11 – Business process with two activities in non-conformance to the acceptable risk level.

in Table 4.3 A constraint is defined for each PEF in the business process that strives to satisfy the non-conformance of this potential execution flow to the acceptable risk level. An example of variables and constraints of a CSP, which refer to the PEF $F_1$, is given in Figure 4.12.

The variables $Risk_{A_1}$, $Risk_{A_2}$, and $Risk_{F_1}$ are defined with an open domain, since, prior to solving the CSP, it is impossible to determine their values until an attempt is made to satisfy the constraints. Subsequently, there are three types of constraints:

1. Risk estimation of activities that defines how the risk determination is calculated based on the risk method previously selected.

2. Risk estimation of PEFs that represents the combination of risks of activities following the patterns defined in Table 4.3.

3. Objectives that represent the achievement of the search of a constraint program (such as defined in Definition 2.3.2).

For instance, in Figure 4.12 the objective constraint $f_1$ is a Boolean constraint that indicates whether the constraint can be satisfied. That is, $f_1$ is evaluated as true iff $Risk_{F_1}$ can achieve a value less than the Acceptable risk $Bp_i$, otherwise $f_1$ is false. When $f_1$ cannot be satisfied, then a non-conformance in this potential execution flow is indicated. In this particular case, the non-conformance consists of the impossibility of satisfying the
Variables: {Integrity_{A1}: [1,3], Integrity_{A2}: [1,5], Frequency_{R1}: [2,4], Consequence_{R1}: [4,5], Frequency_{R2}: [1,3], Consequence_{R2}: [4,5], Frequency_{R3}: [3,4], Consequence_{R3}: [3,5], RiskReduction_{T1}: [2-10], Acceptable risk_{BPI}: 120, Risk_{A1}: [1, 1000], Risk_{A2}: [1,1000], Risk_{F1}: [1,1000], f1: Boolean}

Constraints: 

% Risk Activity 1
Risk_{A2} = (Integrity_{A1}) * ((Consequence_{R1} - Consequence_{R1}*RiskReduction_{T1})* (Frequency_{R1} - Frequency_{R1}*RiskReduction_{T1}) + (Integrity_{A1}) * (Consequence_{R3} - Consequence_{R3}*RiskReduction_{T1})* (Frequency_{R3} - Frequency_{R3}*RiskReduction_{T1}) + (Consequence_{R2} - Consequence_{R2}*RiskReduction_{T1})* (Frequency_{R2} - Frequency_{R2}*RiskReduction_{T1});

% Risk Activity 2
Risk_{A2} = (Integrity_{A1}) * ((Consequence_{R1} - Consequence_{R1}*RiskReduction_{T1})* (Frequency_{R1} - Frequency_{R1}*RiskReduction_{T1}) + (Integrity_{A1}) * (Consequence_{R3} - Consequence_{R3}*RiskReduction_{T1})* (Frequency_{R3} - Frequency_{R3}*RiskReduction_{T1});

% Risk PEF 1
Risk_{F1} = (Risk_{A1} + Risk_{A2})/2;

% Objective of conformance
f1 = (Acceptable risk_{BPI} < Risk_{F1}); }
constraint by attaining a set of assignments of values that achieve less than the acceptable risk value. In the same way, constraints can be defined for other potential execution flows.

When all PEFs are modelled in the same manner, the objective is to find an assignment that enables the satisfaction of the maximum number of constraints related to the potential execution flows. This problem can be modelled as a Max-CSP. A Max-CSP is defined as follows.

**Definition 4.3.4. Max-CSP.** The Maximal Constraint Satisfaction problem (Max-CSP) is an optimization problem over CSPs whose goal is to allocate an assignment to the variables that satisfies as many constraints as possible.

In this case, every PEF within a business process is modelled as a set of constraints in order to compose a Max-CSP. A Max-CSP is defined in a separate model by each business process. The set of PEFs that cannot be satisfied in the Max-CSP are the PEFs in non-conformance to the acceptable risk level. Similarly, a Max-CSP technique can be used to identify which activities and artifacts within a potential execution flow are in non-conformance to a certain acceptable risk level.

In order to automate the diagnosis, we propose a diagnosis process as shown in Figure 4.13. Firstly, business process models and risk model extensions are transformed to specific CSP models. The CSP models (cf. Max-CSP) are then evaluated in a constraint solver which ascertain whether the model is satisfiable (thus in conformance) or unsatisfiable (thus in non-conformance). With respect to non-conformance, the evaluation of the CSP model is used for the determination of PEF, activities and artifacts that are non-conformance.

Various CSP solvers exist on the market such as ChocoSolver (ChocoSolver 2009), JaCoP (Kuchcinski 2012), COMET (Dynadec Decision Technologies).
4.3. DIAGNOSIS OF NON-CONFORMANCE OF RISKS

2012), and IBM CPLEX (IBM 2012). Although CSP solvers share the same aims, each has a particular syntax and features. In order to render our proposal flexible, agile and general, we have defined a solver-independent approach through transformations. A model-driven approach is based on a set of transformations (Model-to-Model) that enables the automatic translation of business process models, extended with a risk model, into platform-specific Max-CSP programs as shown in Figure 4.14. Those transformations implemented can be consulted in Appendix A. This transformation uses a set of auxiliary functions in order to determine PEFs, reified constraints, risk expressions, and other structures that compose the Max-CSP scripts. Whenever any other platforms require support, we only have to provide a transformation to those platforms.

Moreover, our approach supports the selection of CSP solver features at transformation time. The transformation is parameterized to include information about the CSP solver (syntax, search type, etc.) As mentioned earlier, each CSP solver has some specific features for instance local search is supported by COMET but not by Choco. There are scenarios where local searches are necessary. Therefore, we have provided a mechanism to enable the selection of a CSP solver that enables this type of search and, specifically, which local search must be used. This feature provides high flexibility since we can custom needs in each scenario. Furthermore, our approach is defined to support various risk formulae. In order to achieve this feature, another parameter is included in the transformation which indicates the type of risk formula utilized for the determination of the risks. By means of transformation templates, Max-CSP models can be obtained for particular platforms. These CSP models can be performed for a specific constraint solver. The results returned by the solver constitute the diagnosis of non-conformance.

As a result of this approach, we have implemented an OPBUS tool (Varela-Vaca 2012) that integrates: (1) a transformation engine that enable business process models
CHAPTER 4. DIAGNOSIS OF NON-CONFORMANCE OF RISKS

4.3.4 Implementation of diagnosis of non-conformance of risks

The OPBUS tool is an Eclipse plug-in based on MDA technologies. The OPBUS architecture is depicted in Figure 4.15. The OPBUS tool is composed of a business process modeller, a core of tools and a run-time layer. The modeller is provided with various features, such as: validation modules, transformations, and CSP solvers. The OPBUS tool is developed based on Eclipse RCP (Eclipse [2012a]), Epsilon (Kolovos [2012]), and GMF (Eclipse [2012b]). Epsilon and GMF are utilized to generate the modeller, the transformation scripts, and the validation scripts. Eclipse RCP is employed to integrate other components, such as CSP solvers, and to develop characteristics of a more specific nature, such as property windows.

As mentioned earlier, the OPBUS tool presents a business process modeller which supports the specification of the extension presented in Section 4.2 as shown in Figure 4.16. The modeller enables the users to define a business process model and set up the risk information by means of the Properties tab available (cf. Figure 4.16). The Properties tab is where the users can edit properties of the model, such as the acceptable risk value for a business process within objectives, value of assets, and can include vulnerabilities to activities. The modeller is provided with a set of validation scripts that enables the user to detect possible structural faults, such as livelocks, and starvations, and to check the correctness of time events, conditions, etc. These validation scripts can be consulted in

---

**Figure 4.15** – OPBUS plugin architecture.
4.4. A CASE STUDY

Moreover, the OPBUS environment is equipped with specific features to obtain a direct transformation of business process models into COMET models, as shown in the contextual menu of Figure 4.10. The transformation can be configured through property windows as shown in the figure. On this page, the type of transformation (currently COMET and Choco are supported), type of risk formula (currently MAGERIT and Risk_OPBUS are supported), and type of search (currently EXHAUSTIVE and LOCAL are supported) used are selected previous to any validation or transformation. Figure 4.17 shows a property dialogue where OPBUS is being set up in order to transform the business process model into a COMET model, by using an exhaustive search and an OPBUS risk formula. The OPBUS tool integrates the COMET solver for the automatic solution of CSP models. This figure shows how the transformation is configured to use OPBUS as its risk formula (this formula has been described in previous sections). Nevertheless, other formulae can be configured, such as FMEA and MAGERIT.

Furthermore, the OPBUS plug-in is equipped with a validation option, as indicated in the Edit menu in Figure 4.16. In this case, the validation option performs the following steps in an automatic way:

1. business process models in the current workspace are transformed into CSP models.
2. CSP models are solved by a specific solver.
3. Results are returned to the graphical model.

In validation, risk estimation is carried out and non-conformance of risks is identified. The results are retrieved over the graphical elements of the business process, whereby those elements which are in non-conformance are highlighted in red. The identification of these elements in the model constitutes the completion of the diagnosis.

In the following section, a case study is presented as an illustrative example. This example describes the complete process of obtaining an automatic risk assessment of business processes using the OPBUS approach.

4.4 A Case study

In order to illustrate the utilization of our approach and tools, an case study (shown in Figure 4.18) is developed. The example utilized in this section is a business process that describes the arrival of a new employee to an organization. This is an adaptation of a real business process from the Bonita Open Solution suite of examples (Bonita Soft, 2012). The case consists of a business process of three pools that represent the different departments of the organization: Human Resources, Hiring Manager and IT. Human resource shows a business process that consists of eight activities (rounded rectangles), two gateways and a start and end event (circles). Gateways with the +-symbol are AND: all incoming edges are required to pass through the gateway, and the gateway activates all outgoing edges. The other gateways with the ×-symbol are XOR: one incoming edge can pass through the gateway and one of the outgoing edges is activated as a result. Hiring manager shows a business process that consists of seven activities, four gateways (two
Figure 4.16 – Integrated development of OPBUS environment.
AND and two XOR gateways), and a start and an end event. The IT business process consists of three activities, two AND gateways, a start event, and an end event. In the figure, the activity labels are abbreviations of activity names that are listed in Table 4.4.

The business process models are to be assessed. To this end, security experts attach threat scenarios to the model, as shown in Figure 4.18. Human resources and Hiring manager have an associated threat scenario (Threat Scenario 1) composed of three threats and two countermeasures, while IT has an associated threat scenario (Threat Scenario 2) composed of a single threat. In order to simplify the example, the labels of threats and countermeasures are only identifiers. In the following table, a description of threats and countermeasures is given.

Security analysts include these features in order to carry out the automatic risk assessment and identify any non-conformance. Since the problem is focused on the diagnosis of the non-conformance to an acceptable risk level of a business process, the acceptable risk level for each business process must first be defined.

One of the main challenges to security is its measurement (Barabanov et al., 2011). Most risk management proposals provide their own metrics. These metrics are defined by means of qualitative or quantitative approaches. Nevertheless, stakeholders are held responsible for the choice of either the acceptable range of values (in the case of a qualitative approach) or the acceptable qualitative value (with regard to the global qualitative scale). A good practice using a quantitative approach is to define a mapping of the range of values to a qualitative scale, such as given in (Fang and Marle, 2012). This mapping gives an overview of values which may be considered as levels of low, medium or high risk. Subsequently, business and security stakeholders must agree on the acceptable risk level and introduce it into the business process. One of the main advantages of our proposal is that stakeholders only need to adjust the acceptable risk values in the business processes, and then our tool can automatically diagnose whether business processes could achieve or exceed this level.

The main objective is the identification of any non-conformance, therefore security experts should indicate these objectives for each business process separately, as shown
Figure 4.18 – Business process model of the example.

<table>
<thead>
<tr>
<th>Business Process</th>
<th>Abbreviation</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human Resource</td>
<td>RS</td>
<td>Received Signed contract</td>
</tr>
<tr>
<td></td>
<td>NP</td>
<td>Notify Payroll</td>
</tr>
<tr>
<td></td>
<td>SL</td>
<td>Send welcome Letter</td>
</tr>
<tr>
<td></td>
<td>PN</td>
<td>Prepare New employee packet</td>
</tr>
<tr>
<td></td>
<td>NC</td>
<td>Notify hiring manager of new Contract</td>
</tr>
<tr>
<td></td>
<td>BF</td>
<td>Begin employee File</td>
</tr>
<tr>
<td></td>
<td>PP</td>
<td>Prepare Payroll and benefits</td>
</tr>
<tr>
<td></td>
<td>FW</td>
<td>Finish workspace setup</td>
</tr>
<tr>
<td></td>
<td>RSW</td>
<td>Request Software</td>
</tr>
<tr>
<td></td>
<td>RH</td>
<td>Request Hardware</td>
</tr>
<tr>
<td></td>
<td>CE</td>
<td>Configure E-mail account and intranet</td>
</tr>
<tr>
<td></td>
<td>NCE</td>
<td>Notify Employee of Contract</td>
</tr>
<tr>
<td></td>
<td>DW</td>
<td>Determine Workspace</td>
</tr>
<tr>
<td></td>
<td>REI</td>
<td>Request E-mail and Intranet access</td>
</tr>
<tr>
<td>Hiring Manager</td>
<td>RSH</td>
<td>Request Software and Hardware</td>
</tr>
<tr>
<td></td>
<td>NS</td>
<td>Negotiate Space</td>
</tr>
<tr>
<td></td>
<td>BW</td>
<td>Begin Workspace setup</td>
</tr>
<tr>
<td></td>
<td>WE</td>
<td>Welcome Employee</td>
</tr>
</tbody>
</table>

Table 4.4 – Activity abbreviations for the example.
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R11</td>
<td>Incidents caused by staff</td>
<td>Refers to incidents produced by misuse of resources such as misuse of email access, web access and unauthorized access to systems and/or data.</td>
</tr>
<tr>
<td>R12</td>
<td>System failure or data corruption</td>
<td>Refers to failures produced by flaws in software or hardware components such as software bugs, and a computer crash. A technical failure can be catastrophic if, for example, crucial data is lost on a failed hard drive and no backup copy is available.</td>
</tr>
<tr>
<td>R13</td>
<td>Infrastructure failures</td>
<td>Refers to failures produced in the infrastructure of the organization that interrupts the business process such as the loss of an internet connection can interrupt business.</td>
</tr>
<tr>
<td>R21</td>
<td>Infection by viruses or malicious software</td>
<td>Refers to infections produced by viruses or malicious software that interrupts the correct functioning of the resources (computer, software).</td>
</tr>
<tr>
<td>T11.2.2</td>
<td>Privilege management</td>
<td>Restricts and controls the allocation and use of privileges.</td>
</tr>
<tr>
<td>T11.5.2</td>
<td>User identification and authentication</td>
<td>All users should have a unique identifier for their own exclusive use.</td>
</tr>
</tbody>
</table>

Table 4.5 – Description of threats and countermeasures.
CHAPTER 4. DIAGNOSIS OF NON-CONFORMANCE OF RISKS

<table>
<thead>
<tr>
<th>Pool</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human Resources (HR)</td>
<td>{Acceptablerisk : 150} &amp; Risk_{HR} ≤ Acceptablerisk</td>
</tr>
<tr>
<td>Hiring Manager (HM)</td>
<td>{Acceptablerisk : 150} &amp; Risk_{HM} ≤ Acceptablerisk</td>
</tr>
<tr>
<td>IT</td>
<td>{Acceptablerisk : 100} &amp; Risk_{IT} ≤ Acceptablerisk</td>
</tr>
</tbody>
</table>

Table 4.6 – Acceptable risk threshold for business processes.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Threats</th>
<th>Asset Value (Integrity)</th>
<th>Asset Value (Confidentiality)</th>
<th>Asset Value (Availability)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RS</td>
<td>{R11}</td>
<td>[1-2]</td>
<td>[1-2]</td>
<td>[1-2]</td>
</tr>
<tr>
<td>NP</td>
<td>{R11, R12, R13}</td>
<td>[1-3]</td>
<td>[4-5]</td>
<td>[4-5]</td>
</tr>
<tr>
<td>SL</td>
<td>{R11}</td>
<td>[1,1]</td>
<td>[1-1]</td>
<td>[1,1]</td>
</tr>
<tr>
<td>PN</td>
<td>{R11, R12, R13}</td>
<td>[1-4]</td>
<td>[1,1]</td>
<td>[1-1]</td>
</tr>
<tr>
<td>NC</td>
<td>{R11, R12, R13}</td>
<td>[1-3]</td>
<td>[4,5]</td>
<td>[4-5]</td>
</tr>
<tr>
<td>BF</td>
<td>{R11, R12, R13}</td>
<td>[1-1]</td>
<td>[1,1]</td>
<td>[1-1]</td>
</tr>
<tr>
<td>PP</td>
<td>{R11, R13}</td>
<td>[1-4]</td>
<td>[1-4]</td>
<td>[1-4]</td>
</tr>
<tr>
<td>FW</td>
<td>{R13}</td>
<td>[1-4]</td>
<td>[1-4]</td>
<td>[1-4]</td>
</tr>
<tr>
<td>NCE</td>
<td>{R11}</td>
<td>[1-1]</td>
<td>5-5</td>
<td>5-5</td>
</tr>
<tr>
<td>DW</td>
<td>{R11, R12}</td>
<td>[1-1]</td>
<td>[2-2]</td>
<td>[2-2]</td>
</tr>
<tr>
<td>REI</td>
<td>{R11, R12}</td>
<td>[3-3]</td>
<td>[4,4]</td>
<td>[4-4]</td>
</tr>
<tr>
<td>RSH</td>
<td>{R11, R12}</td>
<td>[3-3]</td>
<td>[4-4]</td>
<td>[4-4]</td>
</tr>
<tr>
<td>NS</td>
<td>{R11}</td>
<td>[1-1]</td>
<td>2-2</td>
<td>2-2</td>
</tr>
<tr>
<td>BW</td>
<td>{R11, R12}</td>
<td>[1-1]</td>
<td>[2-2]</td>
<td>[2-2]</td>
</tr>
<tr>
<td>WE</td>
<td>{R11}</td>
<td>[3-3]</td>
<td>[4-4]</td>
<td>[4-4]</td>
</tr>
<tr>
<td>RSW</td>
<td>{R21}</td>
<td>[1-3]</td>
<td>[1-3]</td>
<td>[1-3]</td>
</tr>
<tr>
<td>RH</td>
<td>{R21}</td>
<td>[1-3]</td>
<td>[1-3]</td>
<td>[1-3]</td>
</tr>
<tr>
<td>CE</td>
<td>{R21}</td>
<td>[1-3]</td>
<td>[3-5]</td>
<td>[3-5]</td>
</tr>
</tbody>
</table>

Table 4.7 – List of threats and values associated to the activities of the business process.

In Table 4.6 In the same table, business objectives are established that state that the business process risk cannot exceed the acceptable risk level.

In a second step, business and security experts have to collaborate in order to evaluate the business process activities. The business process activities are configured by linking threats and evaluation to each activity. The relation of threats and value are given in Table 4.7. The asset values are specified by means of intervals of values that indicate the minimum and maximum allowed values. In the same way, properties for threats and countermeasures are described in Table 4.8 and Table 4.9 respectively.

At this point, the security analyst is ready to apply the automatic diagnosis. To this end, the security analyst has to utilize the validation options provided by the OPBUS tools. This option automatically transforms the business process models into CSP models, and a CSP solver obtains the solutions for these constraint models. As explained in the previous section, the diagnosis deals with the identification of which potential execution flows in the business process are in non-conformance to the objectives. In Figure 4.19, the potential execution flows for the example are highlighted. These potential execution flows
Threat Frequency Consequence
R11 [1-3] [2-5]
R12 [3-4] [4-5]
R13 [1-4] [1-4]
R21 [2-3] [2-3]

Table 4.8 – Frequencies and consequences related to threats.

<table>
<thead>
<tr>
<th>Countermeasure</th>
<th>Risk Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>T11.2.2</td>
<td>[10-30]</td>
</tr>
<tr>
<td>T11.5.2</td>
<td>[20-50]</td>
</tr>
</tbody>
</table>

Table 4.9 – Risk Reduction values related to countermeasures.

flows are taken into consideration in the definition of the CSP model.

Since, the OPBUS tools support the transformation into COMET© CSP models, a CSP model is generated for each business process. As explained in the diagnosis section, a Max-CSP model is generated for each business process. Max-CSP models contain the code where variables, domain and constraints are defined. Figure 4.20 shows a piece of COMET© code generated for the Human Resources process. In this example, RS variables and domain (range) are defined. In the same way, there are certain constraints that describe how risk is computed for RS and other constraints in order to verify the conformance of the potential execution flow.

CSP solvers can retrieve more than one solution. In this case, partial solutions indicate possible non-conformances in the potential execution flows. Table 4.10 shows three partial solutions obtained in the solution of the CSP. In the table, solutions are indicated as C or NC, where C indicates that the potential execution workflow shows Conformance to the acceptable risk level and NC indicates Non-Conformance. In order to obtain a global solution, a union of the partial solutions is carried out. Table 4.10 shows an example of a global solution obtained by the union of partial solutions obtained for the business process Human Resource. In this case, the potential execution flows affected by non-conformance are F1, F2 and F3. Therefore, only those activities involved within these potential execution flows are responsible for the non-conformance.

The results obtained for the example are given in Table 4.11. This table shows the results of non-conformance for each potential execution flow separated into business processes.

Table 4.12 presents the risk value obtained in the solution of CSPs for all the ac-
Figure 4.19 – Potential execution flows of business processes.

<table>
<thead>
<tr>
<th>Business process</th>
<th>PEF</th>
<th>Non-conformance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human Resources</td>
<td>F1</td>
<td>NC</td>
</tr>
<tr>
<td></td>
<td>F2</td>
<td>NC</td>
</tr>
<tr>
<td></td>
<td>F3</td>
<td>NC</td>
</tr>
<tr>
<td>Hiring Manager</td>
<td>F1</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>F2</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>F3</td>
<td>NC</td>
</tr>
<tr>
<td></td>
<td>F4</td>
<td>NC</td>
</tr>
<tr>
<td>IT</td>
<td>F1</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>F2</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>F3</td>
<td>NC</td>
</tr>
</tbody>
</table>

Table 4.11 – Results of the diagnosis of potential execution flows.
4.5. RELATED WORK

Some initiatives have emerged in the context of BPM whose aim is to bridge the gap between security and business domains. Most of these initiatives are focused on providing enhancements of business process models through new domain-specific languages for the inclusion of assets, requirements, goals, and threats into business process models. The revised literature has been organized into the following categories:

- Extension of business process models: describes approaches that provide any extension for risk assessment of business process models.
- General approaches that provide any security analysis for business processes.
- Risk treatment for business processes: describes approaches that provide mechanisms for the selection of security controls for business processes.
- Risk evaluation of business processes: describes approaches that enable the identification of non-conformance of risks.

To sum up, a comparison table of the most relevant approaches within the revised literature is given.
Figure 4.21 – Results of the diagnosis of non-conformance in the business process model.
4.5. RELATED WORK

<table>
<thead>
<tr>
<th>Activity</th>
<th>Risk value</th>
<th>Non-conformance</th>
<th>PEF</th>
</tr>
</thead>
<tbody>
<tr>
<td>RS</td>
<td>45</td>
<td>C</td>
<td>{F1,F2, F3}</td>
</tr>
<tr>
<td>NP</td>
<td>243</td>
<td>NC</td>
<td>{F2}</td>
</tr>
<tr>
<td>SL</td>
<td>84</td>
<td>C</td>
<td>{F3}</td>
</tr>
<tr>
<td>NC</td>
<td>473</td>
<td>NC</td>
<td>{F3}</td>
</tr>
<tr>
<td>BF</td>
<td>129</td>
<td>C</td>
<td>{F3}</td>
</tr>
<tr>
<td>PP</td>
<td>248</td>
<td>NC</td>
<td>{F1}</td>
</tr>
<tr>
<td>PN</td>
<td>129</td>
<td>C</td>
<td>{F2}</td>
</tr>
<tr>
<td>FW</td>
<td>160</td>
<td>NC</td>
<td>{F1,F2, F3}</td>
</tr>
<tr>
<td>NCE</td>
<td>150</td>
<td>C</td>
<td>{F3,F4}</td>
</tr>
<tr>
<td>REI</td>
<td>132</td>
<td>C</td>
<td>{F3}</td>
</tr>
<tr>
<td>RSW</td>
<td>132</td>
<td>C</td>
<td>{F4}</td>
</tr>
<tr>
<td>WE</td>
<td>165</td>
<td>NC</td>
<td>{F3,F4}</td>
</tr>
<tr>
<td>RSH</td>
<td>27</td>
<td>C</td>
<td>{F1}</td>
</tr>
<tr>
<td>RH</td>
<td>27</td>
<td>C</td>
<td>{F2}</td>
</tr>
<tr>
<td>CE</td>
<td>108</td>
<td>NC</td>
<td>{F3}</td>
</tr>
</tbody>
</table>

Table 4.12 – Results of the non-conformance by activities.

Extension of business process models

Certain relevant studies provide approaches for the integration of risks into the business process models. In (Cope et al., 2010), a new notation for business process modelling notation (BPMN) is proposed to aid the documentation in risk assessment of business processes. The two major limitations of this approach are: (1) the proposal extension is very complex to understand since it presents three separate models related to the same problem; and, (2) the extension is only valid for BPMN models. In (Rosemann and zur Muehlen, 2005), a taxonomy is proposed to enable the integration of risk into business processes focusing on EPC-based process models. The taxonomy is applied with the aim of enabling the analysis and documentation of business processes. In (Lambert et al., 2006), the authors propose an extension of the integrated definition for function (IDEF) modelling that supports the description of sources of programmatic risk in business processes. In (Churlioiv et al., 2006), a framework for the evaluation of risks in business process management based on value-focused process engineering is presented. In (Rodríguez et al., 2006), an extension to UML 2.0 activity diagrams is proposed so that graphical annotations can be provided for the specification of security requirements in the diagrams. This extension is defined by means of a UML profile called BPSeq. Their approach is very similar to that presented in (Wolter et al., 2009), although in this case Rodriguez et al. provide a set of security annotations to specify requirements in BPMN models. Finally, (Xue et al., 2012) presents an extension of BPMN models with risk management properties.
CHAPTER 4. DIAGNOSIS OF NON-CONFORMANCE OF RISKS

Risk estimation of business processes

Nevertheless, most these approaches lack an automatic risk estimation process that enables the value of risk for business processes to be determined. In (Xue et al., 2012), these authors use the control-flow of business processes to determine the risk in terms of economic consequences of a data error by probabilistic methods. In (Fenz et al., 2009), an approach for the automatic determination of the importance of resources is presented. This approach takes into consideration the control-flow of business processes. The authors use a transformation of BPMN models into Petri-nets and define a formalization to determine the importance of resources used in business processes. In (Neubauer and Heurix, 2008), the authors propose an approach for the risk analysis and the selection of adequate security controls for business processes. However, only an overview of the approach is provided since no details on the estimation of business process risks and on the selection of security controls are given.

General security approaches for business processes

Many more general approaches exist. In (Jürjens, 2002), an UML extension called UMLsec is presented. This extension provides certain UML profiles in order to aid the security-aware development of systems based on UML. Nevertheless, UMLsec is only defined for UML-based developments since it is not a business-oriented approach. Jakoubi et al. propose a notation-independent model as a reference model (Jakoubi and Tjoa, 2009). Likewise, Sackmann (2008) extends current risk management methods by bridging the gap between the business process view and the more technical view of IT risks. These approaches propose theoretical reference models in an attempt to fill the void between business and risk domains. The CORAS method (Lund et al., 2011) conducts context-independent security risk analysis, which is abbreviated to security analysis, and provides a domain-specific language inspired by UML for threat and risk modelling. The CORAS language includes various kinds of diagrams using varied notation, thereby providing a computerized tool designed to support the documenting, maintaining and reporting of analysis results through risk modelling. In (Sienou et al., 2007), a framework that unifies risk management and business process management is presented. This approach is limited to the presentation of various stages of the framework and a description of how it works from a theoretical point of view. In other respects, Neubauer et al. propose a framework for the analysis of the security of business processes from the point of view of cost-benefit (Neubauer et al., 2005). Their framework proposal is defined for integration into any business process management approach.

The main drawback in all these studies lies in the manual nature of their approaches. The lack of tools and mechanisms for the automation of the process of risk assessment in business processes means that these approaches are focused on extending models to enrich their expressivity so that the documentation of risk assessment in business processes can be supported and improved.
4.5. RELATED WORK

Risk treatment for business processes

Other approaches strive towards the automation of the generation of security countermeasures in business process models. In (Menzel et al., 2009), an approach to automate the generation of security controls for business processes in accordance with specific risk thresholds is proposed. The authors provide a risk scale aligned to a set of security controls that can be applied in different parts of the model. Nevertheless, the authors make no previous assessment of the model and hence fail to properly identify which risks exist in the model. Related to this work, (Wolter et al., 2009) propose a set of security annotations for graphical business processes that enable security configurations to be directly set up in the business process model. However, the approach pays no attention to risks or to the previous risk assessment of the model since these authors only focus on presenting the mechanisms to set up and generate specific security configurations in the model. In (Neubauer et al., 2008), the authors focus on the selection of ISO/IEC 27001 controls countermeasures based on multiple objectives (such as cost and benefit). However, only an overview of the approach is provided since no details on the selection of security controls are given.

Automation of risk evaluation for business processes

In the risk assessment, it should be interesting to identify which parts of the business process present non-conformance to the business objectives. To this end, it is desirable to establish a diagnosis technique. Very few contributions deal with diagnosis in business processes. Nevertheless, the majority of diagnosis contributions are focused on identifying faults in business processes. In (Borrego et al., 2009) model-based diagnosis techniques are applied to identify activities that are non-reachable due to design faults. In (Varela-Vaca et al., 2010b) model-based diagnosis techniques are applied to identify structural faults in the design of business processes. In (Varela-Vaca et al., 2011c) the authors use diagnosis techniques to detect faulty components in order to apply a fault tolerance mechanism in the execution of business processes.

Comparison of approaches

In order to give a clear picture of all aforementioned research, a comparative study of the most relevant approaches related to the topic of this approach is provided, following the survey presented in (Jakoubi et al., 2009), as shown in Table 1. This comparison is carried out in accordance with the following categories: (1) Modelling: indicates which modelling languages are supported; (2) Security dimensions: indicates whether the evaluation of assets is carried out with regard to different security dimensions; (3) Cost: indicates whether costs can be considered for the risk assessment; (4) Objectives: indicates whether the approach supports the specification of requirements; (5) Threats and Vulnerabilities (Threats/Vuln.): indicates whether the approach supports the specification of vulnerabilities; (6) Countermeasures (Counterm.): indicates whether the approach supports the specification of Countermeasures; (7) Automatic analysis: indicates whether the ap-
CHAPTER 4. DIAGNOSIS OF NON-CONFORMANCE OF RISKS

... approach supports the automatic assessment of models in order to detect non-conformances. It should be borne in mind that the majority of approaches support several characteristics. Furthermore, not one single approach supports or fosters a process for the analysis of the conformance of the requirements specified with regard to risk or security issues; (8) Risk estimation: indicates whether the approach supports the estimation of the risk value of business processes or specific elements within the model; (9) Control-flow: indicates whether the approach takes into consideration the control flow in the risk determination. The symbol ✓ is used to indicate that the approach supports this category, the symbol ~ is used to indicate that the approach partially supports this category, and the symbol * is used as wildcard to indicate that the approach supports all possible values in this category.

4.6 Summary

In this Chapter, the issue of automatic security risk management in current BPM has been tackled. The main obstacles are related to the lack of awareness in the security risk development of business process-driven products. In the majority of cases, security risk is only considered as an afterthought and risk management is carried out in a manual way after implementation of the working system. We conclude that it is necessary to provide both business and security domain experts with a common model in order to enable capacities of risk assessment and of verification of the conformance of risk with regard to business objectives. A large proportion of the literature revised herein proposes an integration of risk information by extending graphical business process models in order to aid the documentation of business process risk assessments. Several authors propose simple approaches coupled to the business process languages or focused on certain objectives such as time or cost, while others propose overcomplicated graphical extensions. However, none of these studies proposes an integrated mechanism to automate the risk assessment and to check the conformance of these risks.

We have defined an approach which focuses on an risk model extension of business process models and automatic check conformance with regard to objectives of risks. To this end, the proposal uses verification mechanisms based on the model-based diagnosis of which potential execution flows, activities and artifacts are in non-conformance with regard to acceptable risk level. The automation of diagnosis is carried out using artificial intelligence techniques based on constraint programming. The proposal has been supported with the implementation of a plug-in that enables the graphical specification of the extension and the automation of the diagnosis process.
### Table 4.13 – Comparison of approaches

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Cope et al., 2010</td>
<td>BPMN</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Rosemann and zur Muehlen, 2005</td>
<td>EPC</td>
<td>~</td>
<td>~</td>
<td>~</td>
<td>~</td>
<td>~</td>
<td>~</td>
<td>~</td>
<td>~</td>
<td>~</td>
</tr>
<tr>
<td>Lambert et al., 2006</td>
<td>IDEF</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Churiov et al., 2006</td>
<td>EPC</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Rodriguez et al., 2006</td>
<td>UML</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Menzel et al., 2009</td>
<td>BPMN</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Jakoubi and Tjoa, 2009</td>
<td>*</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Neubauer et al., 2005</td>
<td>*</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Sackmann, 2005</td>
<td>Petri-Nets</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Fenz et al., 2009</td>
<td>*</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Neubauer and Heurix, 2008</td>
<td>BPMN</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Xue et al., 2012</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Chapter 5

Automatic Risk Treatment Plans for Business Process Models

Our goal is not to build a platform; it’s to be across all of them.

Mark Zuckerberg

5.1 Context and Motivation

In the previous chapter, we have seen how business process models have been assessed to identify non-conformance of risks with regard to business objectives. That is, a diagnosis process has been performed in business process models extended with risk information in order to identify activities, artifacts, threats, and vulnerabilities that produce the non-conformance of risks. It is therefore crucial for organizations to act as soon as possible by means of selecting adequate risk treatment plans. These threats and vulnerabilities could produce deviations of business processes in the achievement of business objectives. The main problem is therefore to optimally select security controls to be implemented along with business process models that turn non-conformances of risks into conformance in accordance with established business objectives.

As mentioned earlier, the growing trend towards the automation and externalization of business processes by means of Technology Infrastructure (TI), such as Business Process Management Systems (BPMS), has increased the security risks in the organizations. Nevertheless, security in BPMSs is mostly overlooked by default or is taken into consideration in a second phase. Traditionally, security countermeasures correspond to ad-hoc solutions due to late and unexpected threats. In general, there is a lack of mitigation policies or risk treatment plans within organizations.

There exist an enormous number of catalogues, such as those defined by ISO/IEC 27002:2005, COBIT, COSO, HIPAA, and PCI DSS 1.2, that can vary from abstract to specific countermeasures depending on the level of the specification to be implemented (see Figure 5.1). Security managers and administrators are responsible for selecting and
measuring the effectiveness of security controls which comply with the business objectives. However, security countermeasures can vary from the simple installation of antivirus software to the configuration of secure protocols (e.g. https protocol) or even the configuration of an Access Controls List (ACL) for a network firewall. This heterogeneity coupled with the absence of formalism and a huge variability complicate the task of selection and configuration of specific security countermeasures since it implies the involvement of a high level of knowledge and expertise.

The selection and configuration of security countermeasures is one of the main problems within the scope of IT security since, in most cases, it constitutes a human, manual, time-consuming, and error-prone task that involves several security stakeholders, such as security managers and administrators (Davis, 2012). Ideally, this task should be automated in order to reduce the workload for security stakeholders, and to increase the profit of organizations. In (SANS Institute, 2013), a report states that the automation of security controls will substantially reduce the cost of security, while improving its effectiveness.

We propose carrying an analysis of countermeasures to define a catalogue of IT security controls to engage certain security goals in BPMSs. Firstly, in Section 5.2, the problem of the selection of countermeasures in business processes is formalized. Subsequently, we propose a formal model for the representation of countermeasures in business processes based on security patterns and feature models. Furthermore, we propose an enhancement of the representation of patterns and the formalization of feature models with metrics that enable to be inferred the effectiveness and suitability of these countermeasures with regard to the needs of organizations. Automated analysis of feature models, such as Feature-Oriented Domain Analysis (FODA), enables the inference and reasoning capabilities in the form of the selection of components, selection and derivation of parameter value, layout of the selected components, etc. Finally, in Section 5.4, we provide implementations based on FODA and constraint programming techniques for the
5.2 Formal Approach for Security-Risk Treatment Problem

Following the formalization of diagnosis of non-conformances of risks introduced in Chapter 4, we assume a set of $n$ business process models $\{BP_1, BP_2, \ldots, BP_n\}$ of a BPMS that have been assessed. We have identified a set of activities and artifacts of each business process $BP_k$ that must be treated for each security dimension. In Figure 5.2 an example of diagnosis of a BPMS is illustrated. For instance, the diagnosis (represented by $\Delta_{BP_2}$ in the figure) of business process $BP_2$ is $R_{A_1}$ and $R_{A_2}$, where $R_{A_1}$ is only affected in the dimensions of confidentiality and availability, while $R_{A_2}$ is affected in authentication and availability. $\Delta$ represents the diagnosis of all business processes in the BPMS.

In listing 5.1 an example of $\Delta_{BP_k}$ is represented in matrix notation, where the symbol $\sim$ has been used to indicate the dimensions omitted. For instance, the dimensions to be treated for $R_{A_i}$ are integrity, and authentication, while the remaining dimensions are omitted. In this case we have represented the dimensions of integrity (i), confidentiality (c), availability (a), authorization (az), and authentication (at). However other dimensions can be considered.
\[ \Delta_{BP} = \begin{pmatrix} R_{A_0} \\ R_{A_1} \\ \vdots \\ R_{A_m} \end{pmatrix} = \begin{pmatrix} R^i_{A_0} & \sim & \sim & R^c_{A_1} & \sim \\ R^i_{A_1} & \sim & R^c_{A_1} & \sim & R^a_{A_2} & \sim \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ R^i_{A_j} & \sim & \sim & \sim & \sim \end{pmatrix} \] (5.1)

At this point, the BPMS has been diagnosed by identifying the \( \Delta \) responsible for the non-conformance of risk. The application of a risk treatment is proposed in order to mitigate these problems. We strive to determine a set of countermeasures \( SC \) that treat \( \Delta \). Hereinafter, it is assumed that, given a \( R_{BPMS} \) and \( \Delta \), the risk treatment problem can be formalized as follows:

**Definition 5.2.1. Risk-treatment Problem.** A risk-treatment problem is defined by the tuple \((SC(x), \Delta)\) where \( SC(x) \) is the set of countermeasures to be applied in \( x \in \Delta \), and \( \Delta \) represents the activities to be treated.

Therefore, the risk treatment problem becomes the identification of the set of countermeasures, \( SC(x) \), that enables the business goals \( BG \) to be satisfied:

\[(R_{BPMS} - \Delta) \cup SC(\Delta) \cup BG \vdash \top \] (5.2)

Solving a risk treatment problem for a particular \( \Delta \), a set of countermeasures is assumed, denoted as \( SC(x) \).

**Definition 5.2.2. Countermeasure.** A countermeasure, \( SC(x) \), is a security configuration that produces a risk reduction or mitigation in the dimension \( d \) of \( x \in \Delta \), where \( d \in \{i, c, a, at, az\} \).

For a better understanding, an illustrative example is given. Since business goals indicate that the risk in any dimension cannot exceed 15, let us consider \( \Delta \) as given in listing 5.3. It is assumed that the risks and countermeasures are assigned to appropriate quantitative values. The estimation of these values relies on the expertise of various stakeholders and their experience [Neubauer and Heurix 2008].

\[ \Delta = \begin{pmatrix} R_{A_0} \\ R_{A_1} \\ \vdots \\ R_{A_m} \end{pmatrix} = \begin{pmatrix} R^i_{A_0} & \sim & \sim & \sim & \sim \\ R^i_{A_1} & \sim & R^c_{A_1} & \sim & R^a_{A_2} & \sim \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ R^i_{A_j} & \sim & \sim & \sim & \sim \end{pmatrix} = \begin{pmatrix} 15 & \sim & \sim & \sim & \sim \\ 17 & 16 & \sim & \sim & \sim \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 18 & \sim & \sim & \sim & 18 \end{pmatrix} \] (5.3)

For the diagnosis problem above, a set of security configurations \( SC(x) \), is selected, such as that given in listing 5.4. The configuration \( SC^i(x) \) modifies the dimension of the integrity risk of any activity. In this example, it can be observed that: in the first
row, a 0.50 value that represents a 50 percent reduction is applied in the dimension of integrity; in the second row, the 0.50 represents the percentage (50%) of risk reduction applied to integrity for $R_{iA}^i$; 0.60 indicates the percentage (60%) of risk reduction applied to confidentiality for $R_{cA}^i$ and so on.

\[
SC(x) = \begin{pmatrix}
0.50 & \sim & \sim & \sim & \sim \\
0.50 & 0.60 & \sim & \sim & \sim \\
\sim & \sim & 0.60 & \sim & \sim \\
\sim & \sim & \sim & 0.60 & \sim \\
\sim & \sim & \sim & \sim & 0.50
\end{pmatrix}
\] (5.4)

In order to illustrate the calculation of $SC(\Delta)$ the operator $\otimes$ has been defined to represent a matrix, such that for $R = A \otimes B$, the entry $r_{ij}$ is the result of multiplying the entry $a_{ij}$ of $A$ by $b_{ij}$ of $B$. For instance, in listing [5.3] the entry 7.5 is the result of multiplying the (1,1)th entry of $\Delta$ by the (1,1)th entry of $SC(x)$. The final result is a matrix that indicates the new values for $\Delta$. In this case, the risk values comply with the limitations set out in the business goals which state that risk values must not exceed 15.

\[
SC(\Delta) = \begin{pmatrix}
15 & \sim & \sim & \sim & \sim \\
17 & 16 & \sim & \sim & \sim \\
\sim & \sim & 16 & \sim & \sim \\
\sim & \sim & \sim & 18 & \sim \\
\sim & \sim & \sim & \sim & 18
\end{pmatrix} \otimes \begin{pmatrix}
0.50 & \sim & \sim & \sim & \sim \\
0.50 & 0.60 & \sim & \sim & \sim \\
\sim & \sim & 0.60 & \sim & \sim \\
\sim & \sim & \sim & 0.50 & \sim \\
\sim & \sim & \sim & \sim & 0.50
\end{pmatrix} = \begin{pmatrix}
7.50 & \sim & \sim & \sim & \sim \\
8.50 & 9.60 & \sim & \sim & \sim \\
\sim & \sim & 9.60 & \sim & \sim \\
\sim & \sim & \sim & 10.80 & \sim \\
\sim & \sim & \sim & \sim & 9.50
\end{pmatrix}
\] (5.5)

In Figure [5.3] the result of applying countermeasures $SC(x)$ to $\Delta$ is illustrated. The result, $SC(\Delta)$, is the same BPMS whose activities have a set of countermeasures that mitigate the risk in the dimension affected. For instance, activity $A_2$ within $BP_1$ has two countermeasures, one for the integrity ($SC^i$), and the other for the confidentiality ($SC^c$).

On the other hand, the suitability of the configurations, with respect to the requirements defined by the organization, must be measured. In general, organizations might be interested to measure, for instance, cost-benefit, profit, or Return Of Security Investment (ROSI). To this end, metrics could be related to features of countermeasures in order to measure costs, delays, etc. Definition [5.2.1] can therefore be reformulated as follows:

**Definition 5.2.3.** A risk-treatment problem is defined by the tuple $(SC(x), M, \Delta)$, where: $SC(x)$ is the set of countermeasures to be applied; $\Delta$ represents the activities to be treated; and $M$ are the metrics to be considered related to countermeasures.

In listing [5.6] there is an example of costs related to the countermeasures. For instance, the entry 100 indicates the cost in monetary units of the configuration selected for $R_{iA}^i$.

\[
M_{cost} = \begin{pmatrix}
100 & \sim & \sim & \sim & \sim \\
250 & 350 & \sim & \sim & \sim \\
\sim & \sim & 500 & \sim & \sim \\
\sim & \sim & \sim & 1000 & \sim \\
\sim & \sim & \sim & \sim & 600
\end{pmatrix}
\] (5.6)
In most cases, it could prove interesting to find the best configuration according to certain objective functions. In this case, an optimization function is required in order to determine the best solution with regard to the metrics $M$ (i.e. cost, time of delay, etc.). Hence, the problem is reformulated yet again to determine risk treatment problems that optimize certain objective functions.

**Definition 5.2.4. Optimal Risk-treatment Problem.** An optimal risk-treatment problem with a single objective is defined by the tuple $(F, SC(x), M, \triangle)$ where: $SC(x)$ is the set of countermeasures to be applied; $M$ is the set of metrics to be considered; $F$ is the objective function (in the literature denoted by $\text{MAX}$ or $\text{MIN}$); and $\triangle$ represents the activities to be treated, such that:

$$F \equiv \text{MIN}(M_{SC(\triangle)}) \text{or} \text{MAX}(M_{SC(\triangle)}) \quad (5.7)$$

subject to,

$$(R_{BPMS} - \triangle) \cup SC(\triangle) \cup BG \vdash \top \quad (5.8)$$

Thus, the problems consist of the identification of configurations that optimize an objective function. In the following listings, an example of the selection of configurations is presented in which the objective function minimizes the cost. There are two sets of configurations, $SC^1(x)$ and $SC^2(x)$, whose costs are $M_{cost}^{SC^1}$ and $M_{cost}^{SC^2}$, respectively. The application of the objective function results in a new set of countermeasures $SC'(x)$ adjusted to the minimum cost. For instance, for the (1,1)th entry for $SC^1(x)$ and $SC^2(x)$,
the best countermeasure is that given in $SC^1(x)$, since it presents a better risk reduction, although it is more expensive than $SC^2(x)$. In this case, the configurations of $SC^2(x)$ have been selected since they minimize the cost (objective function).

$$SC^1(x) = \begin{pmatrix} 0.50 & \cdots & \cdots & \cdots & \cdots \\ 0.50 & 0.60 & \cdots & \cdots & \cdots \\ \cdots & \cdots & 0.60 & \cdots & \cdots \\ \cdots & \cdots & \cdots & 0.50 \end{pmatrix}, \quad SC^2(x) = \begin{pmatrix} 0.50 & \cdots & \cdots & \cdots & \cdots \\ 0.50 & 0.70 & \cdots & \cdots & \cdots \\ \cdots & \cdots & 0.80 & \cdots & \cdots \\ \cdots & \cdots & \cdots & 0.60 \end{pmatrix}$$

(5.9)

$$M^{SC^1}_{cost} = \begin{pmatrix} 300 & \cdots & \cdots & \cdots & \cdots \\ 250 & 350 & \cdots & \cdots & \cdots \\ \cdots & \cdots & 500 & \cdots & \cdots \\ \cdots & \cdots & \cdots & 1000 & \cdots \\ \cdots & \cdots & \cdots & \cdots & 600 \end{pmatrix}, \quad M^{SC^2}_{cost} = \begin{pmatrix} 200 & \cdots & \cdots & \cdots & \cdots \\ 250 & 450 & \cdots & \cdots & \cdots \\ \cdots & \cdots & 600 & \cdots & \cdots \\ \cdots & \cdots & \cdots & 800 & \cdots \\ \cdots & \cdots & \cdots & \cdots & 650 \end{pmatrix}$$

(5.10)

$$MIN(\{M^{SC^1}_{cost}, M^{SC^2}_{cost}\}) = \begin{pmatrix} 0.50 & \cdots & \cdots & \cdots & \cdots \\ 0.50 & 0.60 & \cdots & \cdots & \cdots \\ \cdots & \cdots & 0.60 & \cdots & \cdots \\ \cdots & \cdots & \cdots & 0.30 & \cdots \\ \cdots & \cdots & \cdots & \cdots & 0.50 \end{pmatrix}$$

(5.11)

In particular cases, it is crucial to find the best configurations according to multiple objectives, such as cost, functionality, and performance. The problem lies in the optimization of searching within a set of countermeasures that complies with $G$ objectives. The problem is reformulated to determine the set of countermeasures that optimize multiple-objective functions.

**Definition 5.2.5. Optimal Risk-treatment Problem with Multiple Objectives.**

An optimal risk-treatment problem with multiple objectives is defined by the tuple $(FG, SC(x), M, \triangle)$ where: $SC(x)$ is the set of countermeasures; $M$ are the set of metrics to be considered; $FG$ is a set of optimization functions $\{F_{g_1}, F_{g_2}, \cdots, F_{g_n}\}$, (in the literature indicated by $MAX$ or $MIN$); and $\triangle$ represents the activities to be treated.

$$FG \equiv MIN(M_{SC(\triangle)}) or MAX(M_{SC(\triangle)})$$

(5.12)

$$FG = \bigcup_i F_{g_i}$$

(5.13)

subject to,

$$(R_{BPMS} - \triangle) \cup SC(\triangle) \cup BG \vdash T$$

(5.14)
As mentioned earlier, one of main problems in the treatment of risks is the selection of countermeasures, \( SC(x) \), to be applied to those activities with a problem of risks, \( \Delta \), that make the problem satisfiable. The selection of a set of countermeasures that are suitable for a set of risks is a \( NP\text{-hard} \) problem similar to that presented in [Zo et al., 2010]. Occasionally, it is possible to find cases that fail to reach a configuration that satisfies all the constraints. In these cases, it is mandatory to report to the different stakeholders that there is no configuration that reduces the risks to an acceptable level. However, stakeholders may choose to implement fault tolerance, such as that proposed in Chapter 6, as a countermeasure against these uncontrolled risks, in order to ensure the execution of business processes.

The aim of our approach is to provide mechanisms for the automatic selection and generation of security configurations for BPMS in order to force business processes to comply with business goals which rely on security risks. To this end, we have proposed an approach based on security patterns and FODA. Firstly, we have provided a flexible model based on security patterns in order to formalize security countermeasures. These security patterns have been enriched with feature models that enable the automated analysis, inference and reasoning of configurations. Subsequently, we have carried out an analysis of the features of the main countermeasures (IT security countermeasures) at the infrastructure level of BPMS. This study resulted in the definition of a set (see Section 5.4) of feature models with core features that can be applied in several dimensions: confidentiality \( (SC^c) \), authorization \( (SC^az) \), authentication \( (SC^at) \), integrity \( (SC^i) \), and availability \( (SC^a) \). Furthermore, in Section 5.5 we introduce feature-oriented domain analysis methods that are used in the implementation to generate configurations in a flexible, agile, and automatic way. In the following section, concepts and formalisms of countermeasures and feature-oriented domain analysis are introduced.

### 5.3 Modelling Security Countermeasures

As introduced in previous sections, one of the main problems in risk treatment is how to describe and/or model countermeasures. We have proposed employing security patterns in the representation and modelling of countermeasures since security patterns (Schumacher et al., 2006) are a widely recognized means for the description of security solutions. Nevertheless, security patterns are usually defined in a textual, informal way, using natural language. In the following section, basic concepts of security patterns are introduced and a formalization of security patterns is given.

#### 5.3.1 Basic Concepts of Security Patterns

Security patterns are based on the idea of design patterns that were introduced by Christopher Alexander in 1977: A pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that pattern.

In general, security patterns, as defined by Schumacher et al., are described in documents that have a specific structure:

- **Name** identifies and summarizes the pattern.
5.3. MODELLING SECURITY COUNTERMEASURES

- **Context** describes the environment before the application of this pattern.
- **Forces** are conditions within the context.
- **Problem** describes a problem that occurs within the context.
- **Solution** describes how to solve problems within the context.

These constitute the mandatory elements; however, other sections, such as dependencies with other patterns, implementation details, structure, examples, can be incorporated into the security patterns in order to improve the information.

In general, security patterns (Schumacher, 2003) are defined in an informal way, using the natural language. Microsoft provides a library of patterns for Security in Web Services (Microsoft, 2012). Nevertheless, these patterns are presented in a textual and informal way. In (Nagappan, 2012), a catalogue of security patterns is defined for security identity and service provisioning in J2EE applications. These patterns are focused on the implementation aspects of patterns. The formalization of the security pattern has been addressed in Schumacher (2003). The authors propose the definition of a knowledge database using security pattern ontologies that enable security pattern concepts and security concepts to be linked. This approach strives to enable search engines and query capabilities. In (Schumacher et al., 2006), a catalogue of security patterns is defined using natural language and UML diagrams. However, these formalizations are unsuitable for the automation of the application of security patterns. On the other hand, in (Menzel, 2010), security patterns have been formalized as profiles to automate the generation of security policies for SOA environments. Menzel provides various security patterns in order to fulfill certain security intentions such as User Authentication, Identity Provisioning, Data Confidentiality, and Data Authenticity. Furthermore, these security patterns have been formalized by means of ontologies and pseudo-formal grammar.

In the following section, our proposal for the modelling of security patterns to specify security countermeasures is given in detail. This model has been unified with the risk model (cf. Figure 4.3, proposed in Chapter 4).

### 5.3.2 Modelling Security Patterns

In our approach, the security patterns are utilized for the selection and generation of countermeasures for certain risks. Hence, the proposed security pattern model must introduce new capabilities with concepts and formalisms for the enhancement of inference in an automatic way. We propose a security pattern meta-model such as that shown in Figure 5.4, whereby security pattern concerns have been related with the risk model proposed in the previous chapter.

This extension strives to represent countermeasures (cf. Countermeasure in Figure 5.4) as security patterns. Similarly, the security pattern meta-model has been related to the risk model by means of red-relations. There are two main relations: (1) Problem are related with vulnerabilities of the risk model; and (2) Security goals and intentions established by a security pattern are related to business objectives defined in the business process through the risk model. For a better understanding of how security patterns
### Security Pattern Meta-Model

<table>
<thead>
<tr>
<th><strong>Risk Model</strong></th>
<th><strong>Business Motivation Model</strong></th>
<th><strong>UML Profile for QoS and FT</strong></th>
<th><strong>Security Pattern Meta-Model</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Objective</td>
<td><strong>Threat</strong></td>
<td>Security goal</td>
<td>Security Dimension</td>
</tr>
<tr>
<td></td>
<td>Threat Scenarios</td>
<td>Security intention</td>
<td>Confidentiality</td>
</tr>
<tr>
<td></td>
<td>Vulnerability</td>
<td></td>
<td>Availability</td>
</tr>
<tr>
<td></td>
<td>Treatment</td>
<td></td>
<td>Authorization</td>
</tr>
<tr>
<td></td>
<td>Countermeasure</td>
<td></td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>Risk Reduction</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Risk Dimension</td>
<td></td>
<td>MIN</td>
</tr>
<tr>
<td></td>
<td>Risk</td>
<td>Expression</td>
<td>MAX</td>
</tr>
<tr>
<td></td>
<td>Risk Value</td>
<td></td>
<td>Constraint</td>
</tr>
<tr>
<td></td>
<td>Risk Criterion</td>
<td></td>
<td>MIN</td>
</tr>
<tr>
<td></td>
<td>Asset</td>
<td></td>
<td>Security Dimension</td>
</tr>
<tr>
<td></td>
<td>Asset Value</td>
<td></td>
<td>Confidentiality</td>
</tr>
<tr>
<td></td>
<td>Dimension</td>
<td></td>
<td>Availability</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Authorization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Figure 5.4 – Security Pattern meta-model.**
5.3. MODELLING SECURITY COUNTERMEASURES

are formalized through the proposal meta-model, its concepts and relations are detailed below.

5.3.2.1 Context

As mentioned earlier, Context describes the environment and its properties before the application of the pattern. We propose formalizing the Context by means of the specification of feature models. Thus, feature models are not only able to define the components within the context, the supported features for the components and their relations, but it also is possible to represent the configurations for the different features of the context. In general, feature models are described in a graphical way; however, constraint languages can be used as a formalism to represent the feature model, such as that shown in Section 2.4. The analysis of feature models also give inference capabilities that enable extra information to be attained, such as that introduced in Section 2.4.

In (Menzel, 2010), a description of Context is disregarded within security patterns. Nevertheless, the author uses ontologies and diagrams as support in the description of Context. For instance, an ontology for Identity Management is given to describe which kind of tokens can be configured as credentials. Each security pattern then defines a diagram to show how the components within the context are linked. For instance, Broken Authentication pattern uses four components: a client, two identity providers (STS), and a service.

A Context example is shown in Table 5.1 where there is an example of a constraint that indicates a condition for protocol SSLv2.

5.3.2.2 Problem

As aforementioned, Problem describes the needs and situations that are produced within Context to provide a solution. In our approach, we have obtained a set of non-conformance of risks related to threats and vulnerabilities that must be treated.

For instance, Direct Authentication (Microsoft, 2012) is a pattern defined by Microsoft where the Problem is described by a sentence such as “How does the Web service verify the credentials that are presented by the client?” In this particular case the Problem is that there exists a need of verification of authenticity of clients in Web Services. This pattern proposes a solution based on direct authentication, that is, the Web service acts as an authentication service to validate credentials from the client. Nevertheless, these descriptions are very textual with a lack of formalization. We therefore propose formalizing Problem through concerns of threats and vulnerabilities.

There exist various vulnerability and threat databases, such as NIST Vulnerability Database (National Institute of Standards and Technology, 2011), and Common Weakness Enumeration (CWE) (CWE, 2011), that provide information referring to technological vulnerabilities. Identifiers utilized by CWE or NIST can be adopted for the specification of security patterns in our approach. For instance, CWE-89 is used to refer to a kind of SQL injection vulnerability in CWE. In addition, CWE provides a particular section (Common Consequences) to indicate which security goal is affected. In the same way,
other databases, such as CRAMM database can be adopted by means of defining the alignment of its concepts.

We propose using identifiers established by CWE (CWE, 2011) for Problem since it provides one of the most relevant and public-access vulnerability database. CWE is currently supported by public and private institutions, such as IBM, Hewlett-Packard, SANS Institute, and Information-technology Promotion Agency, Japan (IPA), etc. In Table 5.1, there is an example of security pattern which Problem is specified using CWE-89 referred to SQL injections vulnerability.

5.3.2.3 Security goals and intentions

We propose the inclusion of new descriptors for Security goal and Security Intention concerns in order to specify new criteria for the selection and inference in security patterns. Security goal is used to refer to a security dimension (e.g. integrity, confidentiality, availability) that the security pattern is enforcing. Security intention is used to describe those aspects related to the security goals. In Table 5.1, there is an example of security goals related to Integrity and a security intention that indicates that the pattern strives towards integrity in data.

In other approaches, such as those approaches in (Menzel, 2010) and (Menzel et al., 2010), the authors define a relation with security intentions. For instance, in (Menzel, 2010), the authors define the security patterns: Secure Pipe and Information Protection as intentions. However it is not clear which security goals the pattern is enforcing, since, a security intention could enforce various security goals, and therefore this relation is also fuzzy.

5.3.2.4 Forces

Forces are conditions and duties within the Context that are necessary or mandatory for the application of the pattern. Therefore, Forces can be defined by constraints classified into two categories: (1) mandatory requirements; and (2) optional requirements. For instance, Direct Authentication pattern in (Microsoft, 2012) defines force such as: ”The Web service can validate credentials from the client against an identity store”. In this example, there is a mandatory requirement that indicates that an identity store is required and that there should exist a direct channel of communication with Web services. In Menzel (2010), Forces is defined by a small grammar with two operators: FORALL and ASSERT. These operators are used to enforce certain conditions with regard to objects of the model.

We propose to formalizing Forces with constraints to represent mandatory (MANDATORY) and optional (OPTIONAL) conditions. Thus, this formalization will enable us to indicates when a feature and a configuration of features must be mandatory or optional. For instance, OPTIONAL(IdentityStore) could represent a constraint that indicates that a feature of an identity store is required. The proposal formalization can be described in the form of BNF grammar as follows:
ForcesExpr ::= Expr OP Expr \hspace{1cm} (5.15)
| Expr | ¬Expr \hspace{1cm} (5.16)

OP ::= BoolOp | RelationOp | MathOp \hspace{1cm} (5.17)

BoolOp ::= ∨ | ∧ \hspace{1cm} (5.18)

Expr ::= ForcesExpr | val \hspace{1cm} (5.19)
| OPTIONAL(Expr) | MANDATORY(Expr) \hspace{1cm} (5.20)

RelationOp ::= < | ≤ | = | > | ≥ | ≠ \hspace{1cm} (5.21)

MathOp ::= + | − | × | ÷ \hspace{1cm} (5.22)

Example of Table 5.1 shows Forces that indicate that Protocol is an optional feature. Thus, the specification of protocol in the security configuration is optional.

5.3.2.5 Solution

Solution should represent mechanisms for the solution of a Problem. For instance, Solution is given in a textual description in Direct Authorization example in [Microsoft, 2012]. On the other hand, Menzel uses a grammar similar to that used in Forces. In that case, Menzel introduces a set of operators such as: REQUIRE, ENFORCE, SET, USE, SCOPE. These operators are used to describe Context configuration.

For our proposal, Solution aims to find the optimal security configuration for a Problem. Thus, the solution is to find a valid configuration within a Context. Therefore, we propose to formalize Solution using two operations:

- **SELECT**(obj) indicates that the goal of the pattern must be the selection of configurations,

- **GENERATE**(obj) indicates that the goal of the pattern must be the generation of configurations,

where **obj** represents a list of objectives in both operations. These objectives are described in Definition 5.2.5. As mentioned in that definition, objective functions are represented by MAX and MIN functions. We propose to use a representation as the following:

- **MAX**(metric), indicates that the objective function is to maximize the value of metric.

- **MIN**(metric), indicates that the objective function is to maximize the value of metric.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context</td>
<td>Constraints</td>
<td>Protocol = SSLv2 → ProtocolCost ≤ 300</td>
</tr>
<tr>
<td>Security goals</td>
<td>String Token</td>
<td>Integrity</td>
</tr>
<tr>
<td>Security intention</td>
<td>String Token</td>
<td>Data integrity</td>
</tr>
<tr>
<td>Problem</td>
<td>String Token</td>
<td>CW-89: SQL injections</td>
</tr>
<tr>
<td>Forces</td>
<td>Constraints</td>
<td>OPTIONAL(Protocol)</td>
</tr>
<tr>
<td>Solution</td>
<td>Constraints</td>
<td>SELECT(MIN(ProtocolCost))</td>
</tr>
</tbody>
</table>

Table 5.1 – Summary of security pattern elements.

In case the of multiple objectives, a list of objectives can be stated. For example, whether the objective selects a configuration such as minimize the \( \text{metric}_1 \) and maximize \( \text{metric}_2 \). This operation can be stated as follows:

\[
\text{SELECT} \{ \text{MIN}(\text{metric}_1), \text{MAX}(\text{metric}_2) \} \quad (5.23)
\]

The aim of our contribution is to achieve the selection and generation of IT security configuration to treat risk detected in business processes, a catalogue of security patterns is presented in the following section. This catalogue is defined to enforce the security dimensions of confidentiality, integrity, authentication, and availability. Firstly, Context of the security pattern is defined through a feature-oriented domain analysis carried out; and subsequently we propose an extension of the feature models with a set of metrics that will be useful for the selection of configurations.

### 5.4 Catalogue of Security Patterns for Business Processes

Business analysts define business process models as mere formalizations, and hence consider them as blueprints. Subsequently, those business process models are given to IT stakeholders for their implementation, thereby rendering IT stakeholders responsible not only for the decision of developing business processes in a specific technology, but also for how these business processes are used, and interconnect and communicate with other systems. As mentioned earlier, business processes are deployed and executed in BPMS that communicate with other systems (Web Services, databases, web application servers, etc). This communication creates a channel of communications between the BPMS where business processes and other systems are executing as shown in Figure 5.5. The example shows the communication between customers, the business processes, and external systems. In most cases, channels must be protected against possible IT security risks (threats or vulnerabilities) since this channel could transport crucial and confidential information. In Figure 5.5, we present the scenario used to illustrate the development of security patterns related to security goals. The infrastructure presented in the scenario is composed of three main components or participants: (1) Customers that transparently use business processes through a proxy and BPMS; (2) Transparent proxy used as a filter
of requests from the customer side and responses from internal requests. In this case, we have used a Load Balancer that redirects external requests to a Web Application Firewall (WAF). A WAF inspects the request against a rule set to seek malicious code. Once the request is accepted by WAF the request is delivered to the corresponding BPMS; (3) BPMS dispatch requests and responses from-to business processes. In the case that a business process needs to use an external resources such as external service, BPMS must request a petition to the Load Balancer and the Load Balancer to Internet and vice versa.

Most leading BPMS today use an infrastructure based on web technology in the form of web application servers with support to web service technologies. For instance, Bonita Soft can be used over Apache Tomcat or JBoss server infrastructure as a deploy server; Intalio BPMS use an Apache ODE BPEL engine; BizAgi supports a variety of servers such as JBoss, Oracle WebLogic and IBM WebSphere. These systems support the configurations and the integration of security countermeasures in many different ways. In this section, we analyse configurations in order to achieve security goals of confidentiality, integrity, availability, authorization, authentication.

According to Definition 5.2.2, the following subsections present a general description of security patterns and a detailed description of the feature analysis carried out to achieve configurations of the dimensions about confidentiality ($SC^c$), authorization ($SC^{az}$), authentication ($SC^{at}$), integrity ($SC^i$), and availability ($SC^a$) in BPMS.

### 5.4.1 Security pattern for Availability and Integrity

A typical vulnerability in development processes is the detection of a specific error, but takes no actions to handle the error. In [CWE-390](https://cwe.mitre.org/index.html), a weakness is defined as *CWE-390: Detection of Error Condition Without Action* that describes this type of vulnerability. *CWE-390* propose various mitigations such as the implementation of testing techniques based on fault tolerance.

We have defined a security pattern in Table 5.2. This pattern aims to achieve security goals of availability and integrity against vulnerabilities related to *CWE-390* by means of the enforcement of fault tolerance mechanisms. *Context* is described by the feature
model $SC^a$ that defines features and relations for the configuration of fault tolerance in executable business processes. We can observe that the security pattern does not provide Solution and Forces sections. These sections are useful to specify the requirements of the organization (e.g. a business objective such as search the fault tolerance that maximize risk reduction, $SELECT(MIN(\text{Risk Reduction}))$).

As previously defined in Definition 2.6.2, fault tolerance strives to preserve the delivery of correct service in the presence of active faults. In accordance with the dependability taxonomy provided by Avizienis (Avizienis et al., 2004), a business process fails when the delivered service deviates from correct service.

Fault tolerance is generally implemented through error detection and subsequent system recovery. The improvement in availability and integrity of business processes thanks to fault tolerance techniques is studied in Section 6. In that section, various fault tolerance patterns are presented that may be integrated into BPMS. These patterns provide mechanisms for error detection and recovery. Furthermore, these techniques present an advantage since they can be integrated independently of the BPMS used.

Proposed fault tolerance patterns vary greatly, and include dynamic binding of services, N-Version Programming (NVP) components, and the check-pointing approach. Each solution requires the configuration of the necessary components. For instance, the dynamic binding solution requires the specification of the number of replicas for each service; the specification of an oracle, which determines whether the solutions obtained are correct; and a binder component, which allows the dynamic binding of services at run-time. Nevertheless, NVP solutions require the selection of a number of variants for services and a kind of adjudicator. The NVP technique does not require an oracle. The complete description of the features and characteristics of these three fault tolerance patterns are given in Table 5.3.

In this case, the feature model is based on those components necessary for the configuration of one of the fault tolerance pattern proposals. For instance, a checkpointing configuration needs, in a first level, the mandatory components of: number of sensors, oracle, and compensation handlers. Nevertheless, the oracle component requires, as mandatory, a diagnoser and a business rule engine. Thus, a valid checkpointing configuration could be as follows:

$$CONF_1 = \{\text{Checkpointing, Number of sensors, Oracle, Compensation handlers, Diagnoser, Business Rules Engine}\}$$  \hspace{1cm} (5.24)

Therefore, this feature model can help to select and to configure a security control with

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem</td>
<td>$CWE-390$: Detection of Error Condition Without Action</td>
</tr>
<tr>
<td>Security goals</td>
<td>Availability, Integrity</td>
</tr>
<tr>
<td>Security intention</td>
<td>Fault Tolerance</td>
</tr>
<tr>
<td>Context</td>
<td>$SC^a$</td>
</tr>
</tbody>
</table>

Table 5.2 – Security pattern for fault tolerance in business processes.
### Feature Dynamic Binding N-Version Programming (NVP) Check Pointing (CP) Description

<table>
<thead>
<tr>
<th>Feature</th>
<th>DB</th>
<th>NVP</th>
<th>CP</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Services</td>
<td>✓</td>
<td></td>
<td></td>
<td>Indicates the number of Web Services used as backup</td>
</tr>
<tr>
<td>(NoS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Binder</td>
<td>✓</td>
<td></td>
<td></td>
<td>Indicates the utilization of a binder component.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Backup Binder</td>
<td>✓</td>
<td></td>
<td></td>
<td>Indicates the utilization of a backup for the binder.</td>
</tr>
<tr>
<td>(BBinder)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oracle</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>Indicates whether an oracle is used. Oracle is composed of a Diagnoser and a Business Rule engine.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Variants</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>Indicates the number of variants used within NV-Components.</td>
</tr>
<tr>
<td>(Nov)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adjudicator</td>
<td>✓</td>
<td></td>
<td></td>
<td>Indicates the kind of adjudicator systems used. Four alternatives are proposed: Exact majority, Median, Mean, and Consensus.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of sensors</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>Indicates the number of check points used.</td>
</tr>
<tr>
<td>(NoSn)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compensation handlers</td>
<td>✓</td>
<td></td>
<td></td>
<td>Indicates the number of compensation handlers used.</td>
</tr>
</tbody>
</table>

Table 5.3 – Description of features for fault tolerance.
respective to the availability in accordance with the fault tolerance patterns. The selection of one or another fault tolerance configuration presents a casuistry that has been captured in the feature model ($SC^a$) of Figure 5.6. The code of the constraint model referring to $SC^a$ can be consulted in the Appendix (cf. Appendix F).

Nevertheless, feature models do not provide metrics to measure how configurations are suitable with regard to certain business objectives such as cost-benefit, risk reduction, ROSI, etc. In the background chapter, we have seen how feature models can be extended with attributes and extra functionalities. In our proposal, the main aim in extending the feature models with attributes and functions is to provide the selection of configurations based on multiple business goals. In this section, for the previously proposed feature model, a set of attributes and functions are given that enable the selection of configurations in accordance with this extension. As a proof of concept, a set of attributes (metrics $M$ indicated in the formalization) are proposed for the feature models of Fault Tolerance and SSL/TLS. In the following section, the proposed attributes and extra functionalities for fault tolerance are set out in detail.

Metrics for feature model of availability and integrity

One of the main challenges in the field of security is the measurement of security [Barabanov et al., 2011]. In general, the units and values employed to determine a risk assessment process are based on statistical data, reports, expertise, and the tracking of assets (Neubauer and Heurix, 2008). In all these cases, these values and units are related to real units. However, there exist cases whereby it is necessary to define abstract scales.

In Table 5.4, we describe the units, range, and description of attributes used to extend the fault tolerance feature model. It must be borne in mind that different scales are applied, depending on the attribute. Certain attributes relate to a real unit (e.g. Time), while for others a generic range of values is used (e.g. Number of Services). Attributes and their relation with feature models are described in Appendix F.

In fault tolerance, a very interesting attribute for the selection could be the Mean Time To Repair ($MTTR$). This attribute is defined as the time required to repair a system failure. In our case, by ignoring the time of detection, the time to repair might be defined as the sum of the delays introduced by each feature. For instance, the $MTTR$ of a dynamic binding solution is the sum of the time of execution of $n$ replicas ($NoR$), the execution of the oracle, and the time of execution of the binders. Attributes $Time$ and $Number$ of $Services$ have been included to enable the calculation of the $MTTR$ for each fault tolerance solution.

\[ MTTR = NoR \times NoR.Time + Oracle.Time + Binder.Time + BBackup.Time \quad (5.25) \]

The values for $MTTR$ introduced in the feature model (cf. Appendix F) were obtained through a performance study of fault tolerance patterns (Varela-Vaca et al., 2011b). In listing 5.26, there is an example that states $NoV$ delay time is between 10 and 30 milliseconds. Thus, the delay time used by variants in an NVP configuration varies from 10 to 30 milliseconds. On the other hand, time of delay used by services in a configuration of
Figure 5.6 – Feature model for fault tolerance.
CHAPTER 5. RISK TREATMENT FOR BUSINESS PROCESS

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Units</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>milliseconds</td>
<td>Integer</td>
<td>Indicates the time of delay introduced for the feature.</td>
</tr>
<tr>
<td>MTTR</td>
<td>milliseconds</td>
<td>Integer</td>
<td>Indicates the mean time to repair a system failure.</td>
</tr>
<tr>
<td>Number of services</td>
<td>Num. services</td>
<td>[0 − 50]</td>
<td>Indicates the number of services, variants, sensors used (NoSN, NVN, NoSN)</td>
</tr>
<tr>
<td>Risk Reduction</td>
<td>%</td>
<td>[0 − 100]</td>
<td>Indicates the percentage of reduction of risk (RR).</td>
</tr>
</tbody>
</table>

Table 5.4 – Description of attributes for the feature model of fault tolerance.

Checkpointing (CP) varies from 10 to 40 milliseconds. Thus, the MTTR of an NVP configuration could be less than the MTTR of a CP configuration depending on the number of variants and services used for each solution.

\[
NVP \rightarrow NVP.NoV.Time \geq 10 \land NVP.NoV.Time \leq 30
\]

\[
CP \rightarrow CP.NoS.Time \geq 10 \land CP.NoS.Time \leq 40
\]

Another attribute included in the extension is risk reduction. As mentioned earlier, the main aim of selecting configurations is to search for configurations that produce a sufficiently large risk reduction to render the risks under the established limit levels. However, it is very complex to measure (in a complete manner) how much risk reduction is produced by a specific countermeasure against certain threats (Barabanov et al., 2011). Our proposal includes the introduction of attributes that aid the estimation of the percentage of risk reduction (Risk Reduction attribute) reached with regard to the Consequence and the Frequency of threats. This information could first be obtained based on the expertise, and subsequently this information could be improved in the revision process in order to obtain better configurations.

For instance, organization requires a configuration with regard to minimal MTTR, thus Solution must be specified with an statement as SELECT(MIN(MTTR)).

5.4.2 Security pattern for Confidentiality, Integrity and Authentication

As mentioned previously, there exist channels of communication between business processes with other systems. CWE (CWE, 2011) describes a vulnerability; CWE-523: Unprotected Transport of Credentials, referring to the application of bad measures to protect credentials in a communication. To ensure the integrity and confidentiality of information exchanged in these communications, a secure channel is mandatory. A secure channel requires the application of digital signatures and encryption infrastructure. This infrastructure can be applied at the transport (SSL/TLS) or message (WS-Security) layer. Secure Socket Layer (SSL) protocol and Transport Layer Security (TLS) protocol (Thomas...
5.4. CATALOGUE OF SECURITY PATTERNS

SSL/TLS are widely used to provide confidentiality, authentication, and integrity in data communications. In fact, CWE propose the enforcement of SSL in the transport layer. We have defined a security pattern in order to provide solutions to the vulnerabilities CWE-523 by means of the enforcement of SSL in the transport layer.

SSL/TLS provides three main security services: confidentiality, by encrypting data; message integrity, by using a message authentication code (MAC); and authentication, through digital signatures and certificates. SSL/TLS allows the authentication of both parties, server authentication with an unauthenticated client, and total anonymity. A detailed analysis of these protocols is given in Appendix D. We have defined a feature model, $SC^{c,i,at}$ that allows the configuration of SSL/TLS.

Most application servers for common web applications employed to deploy business processes, such as Oracle WebLogic, IBM WebSphere, Apache Tomcat, and JBoss, support the establishment of secure channels by means of SSL/TLS. In general, these servers designate connectors that may be set up to use certain providers. There are two main providers: the widely-used JSSE (Java Security Socket Extension) [Oracle 2012], and OpenSSL [Ope, 2012]. However, other providers exist, such as GnuTLS [Gnu, 2012]. These providers support most of the SSL/TLS standards, and further features are available. For instance, GnuTLS supports OpenPGP certificate infrastructure. Application servers enable connectors to be established that use a specific suite of protocols and algorithms. Figure 5.7 shows an example of a configuration of an SSL Socket connector for a Jetty server to use a specific CipherSuite. Other servers, such as Apache Tomcat and JBoss, enable this type of configuration in a similar way.

The configuration of SSL/TLS on these servers is based on the establishment of features such as certificates, certificate authorities, key-stores, ciphers, ports, and protocol versions. A summary of the features considered is given in Table 5.6. Following the standards, connections negotiate a CipherSuite (see details in Appendix V) to be used in the communication. CipherSuite presents an enormous combination of configurations since each Key Change Method can be combined with a number of Cipher and MAC algorithms.

In Appendix, a detailed analysis for the current version of SSL/TLS standards is given. Figure 5.8 shows the resultant feature models ($SC^{c,i,at}$) of the analysis carried out for the current version of the standards and the configuration of SSL/TLS for Oracle WebLogic, IBM WebSphere, Apache Tomcat and JBoss using OpenSSL and JSSE providers. The feature model only shows the values that are supported for each feature, although numerous constraints interrelate these features. Due to the lack of expressivity of feature models, this kind of constraint can be represented as cross-tree constraints. For instance,

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem</td>
<td>CWE-523: Unprotected Transport of Credentials</td>
</tr>
<tr>
<td>Security goals</td>
<td>Confidentiality, Integrity, Authentication</td>
</tr>
<tr>
<td>Security intention</td>
<td>Enfocerment SSL/TLS</td>
</tr>
<tr>
<td>Context</td>
<td>$SC^{c,i,at}$</td>
</tr>
</tbody>
</table>

Table 5.5 – Security pattern for the enforcement of SSL in BPMS.
<Call name="addSSLSocketConnector">
  <Arg>
    <New class="org.mortbay.jetty.security.SslSocketConnector">
      <Set name="Port">8443</Set>
      <Set name="maxIdleTime">60000</Set>
      ...
      <Set name="IncludeCipherSuites">
        <Array type="java.lang.String">
          <Item>TLS_DHE_DSS_WITH_AES_256_CBC_SHA</Item>
          <Item>TLS_DHE_RSA_WITH_AES_256_CBC_SHA</Item>
          <Item>TLS_RSA_WITH_AES_256_CBC_SHA</Item>
          ...
        </Array>
      </Set>
    </New>
  </Arg>
</Call>

Figure 5.7 – Example of SSL/TLS configuration for a Jetty Server

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CipherSuite</strong></td>
<td>Indicates the suites of key change method, cipher and message authentication code are supported.</td>
</tr>
<tr>
<td><strong>Key Change Method</strong></td>
<td>Indicates that cryptographic algorithms have been employed to generate cryptographic keys.</td>
</tr>
<tr>
<td><strong>Cipher</strong></td>
<td>Indicates that conventional cryptographic algorithms are employed to encrypt the message in the transmission.</td>
</tr>
<tr>
<td><strong>Message Authentication Code (MAC)</strong></td>
<td>Indicates the algorithm employed to encrypt the message to provide integrity.</td>
</tr>
<tr>
<td><strong>Protocol</strong></td>
<td>Indicates the version of protocol to be used.</td>
</tr>
<tr>
<td><strong>Session ID</strong></td>
<td>Indicates the ID session established in the negation of client-server connection.</td>
</tr>
<tr>
<td><strong>Authentication Method</strong></td>
<td>Indicates the authentication method to be used (certificates or shared keys).</td>
</tr>
<tr>
<td><strong>Digital Signature</strong></td>
<td>Indicates other types of signature support instead of certificates (SRP, PSK, or Anonymous).</td>
</tr>
<tr>
<td><strong>Certificate</strong></td>
<td>Indicates the type of certificates supported (x509 or OpenPGP).</td>
</tr>
</tbody>
</table>

Table 5.6 – Description of features for SSL/TLS configuration
Figure 5.8 – Feature model of SSL/TLS.
SSL v3.0 introduces Fortezza as a Key Exchange Method, although this method cannot be used with MD5 as a Message Authentication method (MAC). In Appendix D, complete cross-tree constraints related to the FM of SSL/TLS are given.

Finally, it should be borne in mind that it is very hard to separate the configuration of countermeasures for only data integrity or data confidentiality since confidentiality implies the application of integrity and authentication mechanisms. In this section, a feature model for the configuration of three security dimensions is defined. In the analysis of configurations, however, the independence of countermeasures is assumed. Thus, configurations for confidentiality, integrity, and authentication can be selected separately.

In the same way as in fault tolerance, a set of metrics and extra functionalities for $SC^{c,i,at}$ feature model are set out in detail in the next section. The complete code of extended feature model is given in Appendix D.

**Metrics for feature model of enforcement of SSL/TLS**

Various metrics typically used to measure the cost-benefit of security, have been included, as shown in (Sonnenreich et al., 2005). Currently, the most important information for organization is Return on Investment for a security investment ($ROSI$). In general, $ROSI$ is calculated following the formula below (Sonnenreich et al., 2005):

$$
ROSI = \frac{RiskExposure \times RiskMitigation(\%) - Cost}{Cost} \tag{5.28}
$$

Following the proposal (Sonnenreich et al., 2005), in order to estimate the risk of exposure, the Annual Loss Exposure ($ALE$) can be used. $ALE$ is stated by multiplying the projected cost of a security incident (Single Loss Exposure, or $SLE$) with its estimated annual rate of occurrence ($ARO$). There are no standard methods that describe how to measure $SLE$ and $ARO$. As mentioned earlier, this information is estimated based on statistical data, expertise, and the tracking of assets and countermeasures.

$$
ALE = ARO \times SLE \tag{5.29}
$$

Assuming that $SLE$ is fixed, the only way to reduce $ALE$ is by reducing the $ARO$. The reason for the introduction of countermeasures is to reduce the consequence of the frequency of occurrence of threats. Therefore, we propose an attribute; $AROR$, which measures the percentage of $ARO$ mitigation.

$$
ALE = (ARO - AROR(\%)) \times SLE \tag{5.30}
$$

A summary of metrics is given in Table 5.7, where the units, range and description are detailed. The relation metric-features are defined using extra functions. A set of extra functions (functions and constraints) are included in the feature model that are listed in the Appendix.
### Table 5.7 – Description of attributes for feature model of SSL/TLS

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Units</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARO</td>
<td>Times/year</td>
<td>Integer</td>
<td>Indicates the annual rate of occurrence.</td>
</tr>
<tr>
<td>SLE</td>
<td>Monetary units</td>
<td>Integer</td>
<td>Indicates the single loss exposure.</td>
</tr>
<tr>
<td>ALE</td>
<td>Monetary units/year</td>
<td>Integer</td>
<td>Indicates the annual loss exposure.</td>
</tr>
<tr>
<td>AROR</td>
<td>%</td>
<td>[0, 100]</td>
<td>Indicates the percentage of reduction of ARO.</td>
</tr>
<tr>
<td>Cost</td>
<td>Monetary units</td>
<td>Integer</td>
<td>Indicates the cost of implementation.</td>
</tr>
</tbody>
</table>

A pseudo-formal logic is employed to represent these constraints and functions. For instance, the formula of calculation of ALE is only established if the SSL/TLS feature is in the configuration (e.g. \( \text{SSL/TLS} \rightarrow \text{SSL/TLS}_{\text{ALE}} = (\text{SSL/TLS}_{\text{ARO}} - (\text{SSL/TLS}_{\text{ARO}} \times (\text{SSL/TLS}_{\text{AROR}})/100)) \times \text{SSL/TLS}_{\text{SLE}} \)). Furthermore, the symbol \( \neg \) is used to indicate the avoidance of a feature. For instance, the AROR for certificates has to be set to zero in the case of ignoring certificates (i.e. \( \neg \text{SSL/TLS.Certificate} \rightarrow \text{Certificate}_{\text{AROR}} = 0 \)).

ALE for a SSL/TLS configuration is stated by multiplying the ARO, AROR and SLE. The AROR is defined as depending on the Protocol and the infrastructure for Authorization selected (Certificates or DigitalSignatures). The Cost attribute is similarly established. Using these metrics, the ROSI of any configuration can be determined.

On the other hand, other constraints have been stated, such as the constraint that limits ALE to at least 3000 monetary units/year, and the constraint that restricts the range of values for AROR from 20 to 30 in the case of protocol SSLv2.0. Thus, there are constraints to limit the values for attributes, and others constraints are stated to calculate functions related to these attributes. In the following section, a set of initial values is provided as a proof of concept to carry out a comparison of results as close to the reality as possible. In general, the values of these metrics are established based on statistical data, reports, tracking processes and the expertise of professionals.

These metrics enable the suitability of the solution in terms of cost-benefit to be measured. The organization can use the security pattern defined and state a search for a configuration with the minimum ALE or the highest ROSI as possible. This search can be specified using an statement for Solution such as \( \text{SELECT(MIN(ROSI))} \).

### 5.4.3 Security pattern for Authorization

Current business process notations, such as BPMN and UML diagrams, use a role-based perspective. Business process models can therefore define which specific roles are responsible for executing the activities represented in the business processes. In general, business process activities interact with external Web Services. Hence, users/customers enter the BPMS (authentication) taking a specific role, and are enabled to execute the
Traditionally, authorization is established by means of filtering mechanisms (Macfarlane et al., 2012). Filtering mechanisms consist of the specification of security policies (such as RBAC policies) that define conditions (rules) under which users can or cannot access the resources (systems or information). In recent years, Web Application Firewalls (WAF) have emerged which block unauthorized access (Gupta and Saikiah, 2007). These mechanisms inspect the contents of traffic (HTTP traffic), and block specified content, such as access to certain websites, or attempts to exploit known logical vulnerabilities (web attacks by SQL injections or Cross-Site Scripting (XSS) (OWA, 2008)). In the list of the top 25 vulnerabilities, OWASP has CWE-89: SQL injection as the most important kind of vulnerability. On the other hand, cross-site scripting vulnerability is (CWE-79: Cross-site Scripting) listed in the fourth position. We have defined a security pattern for Problems related to vulnerabilities CWE-89 and CWE-79 that enforce authorization by means of a configuration of web application firewalls.

In general, WAF systems are located as full and transparent proxy. In our case, a WAF can be located between the BPMS and other systems and is responsible for the inspection of the content of the request from and response to the BPMS. A WAF can be installed as part of the IT infrastructure by means of appliance systems installed in front of BPMS or as an add-on of the BPMS. Various open-source WAF solutions exist: TrustWave ModSecurity®, QualysIronBee, and OWASPESAPIJava-WAF. Furthermore, other commercial solutions are available, such as AQTronixWebKnight, StingrayApplicationFirewall, WebCastellum, ZionSecured, Guardian@JUMPERZ.NET, and EasyWAF.

WAFs are stated by policies that are composed of a set of rules. If any content matches any rule, then an action is triggered and the web traffic goes through the WAF. Rules can be customized in order to detect common web attacks (typical code injections), malicious web content, malicious file upload, violation of protocols, and Denial of Service attacks (DoS), among others. A summary of the main features for the configuration of rules in WAFs is given in Table 5.9.

The specification of rules presents a highly intricate casuistry. Rules may be defined with regard to certain web content, such as files, response, request information, environment information; operators, such as validation of schemas, verification of schemas, detection of regular expressions; actions, such as allow, block, log content; among others. This complexity requires a high level of knowledge and expertise of the functionalities

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem</td>
<td>CWE-89 - SQL injection</td>
</tr>
<tr>
<td></td>
<td>CWE-79 - Cross-site Scripting</td>
</tr>
<tr>
<td>Security goals</td>
<td>Authorization</td>
</tr>
<tr>
<td>Security intention</td>
<td>Enforcement Authorization</td>
</tr>
<tr>
<td>Context</td>
<td>SCaz</td>
</tr>
</tbody>
</table>

Table 5.8 – Security pattern for the enforcement of Authorization in BPMS.
5.4. CATALOGUE OF SECURITY PATTERNS

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Indicates the data fields against which the operations are applied.</td>
</tr>
<tr>
<td>Operator</td>
<td>Indicates the types of operations to perform against the variables.</td>
</tr>
<tr>
<td>Actions</td>
<td>Indicates the type of action to execute in the case of matching.</td>
</tr>
<tr>
<td>Transformation</td>
<td>Indicates the transformation to apply over data before the application of operators.</td>
</tr>
<tr>
<td>Phase</td>
<td>Indicates when the rule is applicable.</td>
</tr>
</tbody>
</table>

Table 5.9 – Description of features for WAF configuration.

SecMarker BEGIN_HOST_CHECK
SecRule REQUEST_HEADERS:Host "@eq 0" \ "skipAfter:END_HOST_CHECK,phase:2,rev:'2.2.4',t: none, block, msg:' Request Missing a Host Header',id:'960008',tag:'PROTOCOL_VIOLATION/MISSING_HEADER_HOST',tag:'OWASP_TOP_10/A7', tag:'PCI/6.5.10',severity:'5',setvar:'tx.msg=%{rule.msg}', setvar:tx.anomaly_score=+%{txnotice_anomaly_score}, setvar:tx.protocol_violation_score=+%{tx.notice_anomaly_score}, setvar:tx.%{rule.id}-PROTOCOL_VIOLATION/MISSING_HEADER-%{matched_var_name}=%{matched_var}"

Figure 5.9 – Example of rules for the detection of protocol anomalies in ModSecurity.

of the WAF. We have defined a feature model (SCaz) of the most common WAF rule features. This feature model gives useful information for security experts in order to obtain/analyse configurations in accordance with specific and necessary features. We have developed an analysis of the most widely used WAF (see the Appendix). As a result of this analysis, a feature model has been defined as shown in Figure 5.10. This feature model shows only certain features since the representation of all features would render the diagram impossible to decipher.

In this section, various security patterns and feature models (SCn, cat, SCa, and SCaz) have been presented relative to security tools that enable the achievement of security goals for business processes.

Summarizing thus far, we strive towards the best configuration that will enable the risks detected in business processes to be mitigated. To this end, we have proposed a model in order to formalize IT security countermeasures by means of security patterns. This model has been enriched with feature models and metrics in order to provide inference and reasoning mechanisms. We need mechanisms to search for and obtain configurations from feature models. These configurations must be suitable for one or multiple business objectives, such as risk reduction, cost-benefit, and ROSI.

Feature models are analysed automatically by means of feature-oriented model analysis based on constraint programming techniques. In the following section, the analysis
CHAPTER 5. RISK TREATMENT FOR BUSINESS PROCESS

Figure 5.10 – Feature model for WAF.
5.5. AUTOMATIC SELECTION AND GENERATION OF SECURITY CONFIGURATIONS

In order to automate the selection generation of configurations, feature-oriented domain analysis based on Constraint Programming techniques is used. Feature models have therefore been implemented into Constraint Satisfaction Problems (CSP) following the approach explained in Section 2. Feature models have been implemented in COMET© (Dynadec Decision Technologies, 2012). COMET© is a very powerful constraint solver with features for optimized searches. Since SAT solvers are more commonly employed in this kind of analysis due to their promising performance results achieved using the COMET solver, we have decided to carry out the entire analysis using this CSP solver. To the best of our knowledge, COMET© has never been used for any current feature-oriented domain analysis tool, such as FamaSuite (Benavides et al., 2007), VariaMos (Mazo et al., 2012), and pure::variants (Pur, 2012). The constraint programmes used in this section are available for evaluation and downloading at (Varela-Vaca, 2012). These models can be integrated and executed as part of our tools since COMET has been integrated in the plug-in as shown in Chapter 4.

The first analysis consists of the identification of the total number of configurations. The result for this analysis is given in Table 5.10. The analysis indicates: number of features (NF); relations (mandatory, optional, XOR and OR); void feature model (Benavides et al., 2010) is a model validation operation that indicates whether the feature model is void (cf. indicated by •) or not (cf. indicated by ×); number of configurations; and time of performance to obtain the configurations. A feature model is void if it represents no products. The reasons that may make a feature model void are often related with a wrong usage of crosstree constraints. For this analysis, the (exhaustive) default search provided by the constraint solver is applied.

The number of configurations represents the number of all valid configurations that are achieved with this FM. Here, security administrators have to select from among all these configurations; however such a large number of configurations cannot be handled by humans, such as in the case of SSL/TLS (SCi,c,at). Nevertheless, the number of

<table>
<thead>
<tr>
<th>Feature Model</th>
<th>NF</th>
<th>Mandat.</th>
<th>Optional</th>
<th>XOR</th>
<th>Or</th>
<th>Void</th>
<th>Number of conf.</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCa</td>
<td>17</td>
<td>8</td>
<td>1</td>
<td>7</td>
<td>0</td>
<td>×</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>SCi,c,at</td>
<td>49</td>
<td>10</td>
<td>0</td>
<td>42</td>
<td>5</td>
<td>×</td>
<td>3.683</td>
<td>4.699</td>
</tr>
<tr>
<td>SCaz</td>
<td>62</td>
<td>6</td>
<td>6</td>
<td>57</td>
<td>4</td>
<td>×</td>
<td>241.920</td>
<td>77.427</td>
</tr>
</tbody>
</table>

Table 5.10 – Results of feature-oriented domain analysis.
configurations for fault tolerance remains very low. In the worst cases, the number of configurations might be prioritized by means of ordering criteria. It should be borne in mind that the time required to determine the configurations is very low even in the worst case.

As mentioned earlier, feature models can be extended with attributes and extra functionalities in order to adjust the searches. In this case, extended feature models might be transformed into Constraint Optimization Problems (COP), such as those shown in Chapter 2. A COP searches for a solution in accordance with an optimization function, such as that defined in Definition 5.2.4. Thus, COP strives to find the best configuration feasible with an objective function with regard to the attributes included in the FM. For instance, the level of security of an organization can be considered high when the connections between client and server use certificates whose keys are generated using AES algorithms. These types of algorithms are only supported by TLSv1.0 and earlier versions. In this case, the organization can consider the attributes of security level, and is interested in configurations whose security level is high. Therefore, the COP aim would be to search for the best configuration with a high level of security.

In real scenarios, organizations require countermeasures to be selected according to multiple criteria. Therefore, various objective functions are used, for instance cost minimization, and the maximization of the use of resources. In this case, the problem is to determine the configuration based on multiple objectives, such as those defined in Definition 5.2.5. Occasionally, the objectives may be weighted in order to grant one objective
5.5. AUTOMATIC SELECTION OF CONFIGURATIONS

a higher priority than the others. This kind of search is called a weighted search and is a specialization of a multiple-objective search.

In the revised literature, the authors in (Neubauer et al., 2008) (Neubauer and Heurix, 2008) propose a multi-objective approach for the optimal selection of controls. In (Neubauer et al., 2008), the authors propose the selection of ISO/IEC 27001 controls based on multiple objectives, (cost, benefit, etc.). Nevertheless, Neubauer and Heurix merely indicate that selection is carried out using search-based techniques, and that there are no implementations.

In our approach, the generation of optimal configurations is proposed based on multiple objectives by means of Multi-objective COP (MCOP) (Ehrgott and Gandibleux, 2003). In this case, COPs are adapted to attain multiple objectives. To the best of our knowledge no contributions exist in feature-oriented model analysis towards obtaining configurations based on multi-objectives. The transformation from feature models to COP and CSP have been carried out by adapting by hand the code of feature models to search requirements.

Our implementations of feature models with attributes and extra functionalities have been extended in order to generate configurations according to certain optimization criteria. The extension provides information relative to metrics ($M$), such as that indicated in the formalization. Therefore, we have two options for the selection (cf. Figure 5.11): (1) one that provides all possible configurations (referred as Determine all configurations $SELECTION(x)$); and (2) one that provides the best configurations (referred as Determine optimal configuration $SELECT(x)$ according to certain optimization criteria (in the formalization denoted as $F$, and referred as Optimization criteria MIN/MAX in this figure).

Performance Results

In order to compare the results of the determination of configuration with and without optimization, a comparative analysis is given in Table 5.11 which assumes that the constraints and functions are included in the feature models. The comparative study has been performed in two phases: (1) a search using single objectives; (2) a search using multi-objectives. The table shows information on the attributes included: optimization function used for each case; number of configurations achieved; and the performance given in milliseconds. The hardware employed to obtain the results in this section is an Intel Core i7 2.20GHz with 8 GB RAM and Windows 7 Home Premium OS.

As shown in Table 5.11 the $ALE$ is calculated by the combination of $ARO$, $AROR$ and $SLE$. In the cases of single objectives, the constraint solver finds all the configurations (if any). Nevertheless, in the case of multi-objectives, the constraint solver strives to identify the Pareto-efficient combinations. It can be observed that the first search retrieves no solutions in the case of multi-objectives. It is therefore impossible to find a solution that fits the minimum of $Cost$ and $ALE$ due to the problem being over-constrained. In this case, there are two options: (1) accept that there are no solutions; (2) relax some of the objectives in order to find solutions close to the optimum. A multi-objective search for SSL/TLS ($SC_{c,at}$) has been performed, which relaxes the objectives (marked by the symbol $\sim$). In our initial tests, $SLE$ and $ARO$ hold a fixed value, and the $AROR$ is
calculated in terms of the configuration selected. In this case, it can be observed how the minimum has been tightened by eight iterations.

It should be pointed out that the numbers of configurations are high due to the combination introduced by the attributes and extra functionalities. An example of configurations obtained for the SSL/TLS ($SC^{a,c,at}$) example is shown in Table 5.12. This table shows different configurations achieved for the multi-objective of minimizing $ALE$ and relaxing $Cost$. This configuration can help security stakeholders to deal with decision-making regarding security configurations. However, these configurations give an overview of the space of configurations and even specific configurations can be customized with respect to certain multi-objectives. Nevertheless, a large number of configurations remain inoperative for security administrators; it should be interesting, for example, to introduce an ordering criterion in order to generate a list of the best configurations according to the tightened objectives, such as indicated in the # column.

Regarding fault tolerance results, it can be observed that good results are obtained for single-objective searches. In the case of multi-objectives, these solutions are achieved very fast since there are no dependencies between the objectives. Therefore, the constraint solver does not need to find Pareto-efficient solutions since the optimum is determined independently. In Table 59, several configurations are given; the solutions that match with the optimum configurations are highlighted.

Table 5.11 – Analysis of generation of configurations with attributes.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Optimization criteria</th>
<th>Number of conf.</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SC^{a,c,at}$</td>
<td>$\text{SELECT(MIN(ALE))}$</td>
<td>13.138</td>
<td>2,041</td>
</tr>
<tr>
<td></td>
<td>$\text{SELECT(MAX(AROR))}$</td>
<td>5.268</td>
<td>1,255</td>
</tr>
<tr>
<td></td>
<td>$\text{SELECT(MIN(Cost))}$</td>
<td>1.800</td>
<td>2,394</td>
</tr>
<tr>
<td></td>
<td>$\text{SELECT({MAX(AROR), MIN(ALE)})}$</td>
<td>5.268</td>
<td>5,257</td>
</tr>
<tr>
<td></td>
<td>$\text{SELECT({MIN(Cost), MIN(ALE)})}$</td>
<td>0</td>
<td>406</td>
</tr>
<tr>
<td></td>
<td>$\text{SELECT({MIN(Cost), MIN(ALE)})}$</td>
<td>108</td>
<td>880</td>
</tr>
<tr>
<td>$SC^a$</td>
<td>$\text{SELECT(MIN(MTTR))}$</td>
<td>4</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>$\text{SELECT(MAX(Risk Reduction))}$</td>
<td>58</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>$\text{SELECT({MIN(MTTR), MAX(Risk Reduction)})}$</td>
<td>36</td>
<td>39</td>
</tr>
</tbody>
</table>

5.6 Related Work

As mentioned in previous chapters, a number of initiatives have emerged in the context of BPM in order to bridge the gap between risk assessment and business domains: Cope et al. (2010); Lambert et al. (2006); Rodríguez et al. (2006); Rosemann and zur Muehlen (2005). Nevertheless, none of these approaches provides a mechanism for the automation and identification of suitable treatment to mitigate risks.

There exist several contributions where the selection of risk mitigation plans at the Design and Analysis phase is proposed. In Jakoubi et al. (2009), Risk-Oriented Process Evaluation (ROPE) method is proposed. ROPE includes a Counter Measure Sub-Process...
### Table 5.12 – Configurations of SSL/TLS ($SC_{i,c,at}$) for optimization of $ALE$ and $Cost$.

```plaintext
<table>
<thead>
<tr>
<th>#</th>
<th>Digital Signature</th>
<th>Certificate</th>
<th>CipherSuite</th>
<th>Protocol</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>~</td>
<td>~</td>
<td>RSA</td>
<td>~</td>
<td>TLSv1.0</td>
</tr>
<tr>
<td>2</td>
<td>~</td>
<td>~</td>
<td>RSA</td>
<td>~</td>
<td>MD5</td>
</tr>
<tr>
<td>3</td>
<td>~</td>
<td>~</td>
<td>RSA</td>
<td>IDEA-128</td>
<td>SHA-1</td>
</tr>
<tr>
<td>4</td>
<td>~</td>
<td>~</td>
<td>Fortezza</td>
<td>~</td>
<td>SHA-256</td>
</tr>
<tr>
<td>5</td>
<td>~</td>
<td>~</td>
<td>DHE-RSA</td>
<td>3DES-168</td>
<td>SHA-1</td>
</tr>
</tbody>
</table>
```
### Dynamic Binding NVP CheckPointing Objectives

<table>
<thead>
<tr>
<th>#</th>
<th>Replicas</th>
<th>Oracle</th>
<th>Binder</th>
<th>BBinder</th>
<th>NVP</th>
<th>Adjudicator</th>
<th>Sensors</th>
<th>CheckPoints</th>
<th>MTTR</th>
<th>Risk Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>~</td>
<td>~</td>
<td>~</td>
<td>~</td>
<td>5</td>
<td>Variants/10 ms</td>
<td>Median/10 ms</td>
<td>~</td>
<td>~</td>
<td>~</td>
</tr>
<tr>
<td>2</td>
<td>~</td>
<td>~</td>
<td>~</td>
<td>~</td>
<td>7</td>
<td>Variants/10 ms</td>
<td>7 Variants/10 ms</td>
<td>~</td>
<td>~</td>
<td>~</td>
</tr>
<tr>
<td>3</td>
<td>~</td>
<td>~</td>
<td>~</td>
<td>~</td>
<td>9</td>
<td>Variants/10 ms</td>
<td>7 Variants/10 ms</td>
<td>~</td>
<td>~</td>
<td>~</td>
</tr>
<tr>
<td>4</td>
<td>~</td>
<td>~</td>
<td>~</td>
<td>~</td>
<td>~</td>
<td>~</td>
<td>~</td>
<td>2 replicas/10 ms</td>
<td>45 ms</td>
<td>15 ms</td>
</tr>
<tr>
<td>5</td>
<td>10 replicas</td>
<td>10 ms</td>
<td>30 ms</td>
<td>10 ms</td>
<td>10 ms</td>
<td>~</td>
<td>~</td>
<td>~</td>
<td>~</td>
<td>~</td>
</tr>
</tbody>
</table>

**Table 5.13** – Configurations of fault tolerance for optimization of MTTR and Risk Reduction.
where countermeasure actions are defined against threat activities. Nevertheless, this approach lacks a proposal for the automatic and optimal selection of suitable countermeasures. In (Churliov et al., 2006), the authors propose a phase in the proposal for risk mitigation: however, there is no well-accepted formal theory that describes how to carry out this task. In (Asnar and Giorgini, 2006), the authors provide a goal-driven approach as an extension of Tropos/i* in order to analyze risk at organization level. Furthermore, they illustrate a number of different techniques to help the analyst in identifying and enumerating relevant countermeasures for risk mitigation. Nevertheless, the authors focus on studying the selection of countermeasures in cost-effective terms. In (Fenz et al., 2009), the authors propose the selection of ISO/IEC 27001 controls based on multiple objectives, (cost, benefit, etc.). Nevertheless, it is merely indicated that selection is carried out using search-based techniques, and there is no reference to implementations. In (Bai et al., 2007), a probabilistic framework is presented for risk assessment, and an optimization-based approach is proposed to determine the manner in which control procedures can be embedded in the business process to mitigate risk. Nevertheless, it is a theoretical approach that fails to indicate how the optimized selection of controls is carried out. On the other hand, automatic techniques are proposed in (Conforti et al., 2012) to determine mitigation actions at run-time of YAWL business processes. These mitigations are determined by MOSA algorithms that enable the selection of countermeasures by using multiple objectives.

In order to give a clear picture of all the aforementioned research, a comparative study of the most relevant approaches related to the selection of countermeasures are shown in Table 5.14. This comparison is carried out in accordance with the following categories: (1) Design: indicates whether the approach is oriented to be applied at the design stage of business processes; (2) Run-time: indicates whether the approach is oriented to be applied at the run-time stage of business processes; (3) Search: indicates which type of techniques are used to select the countermeasures; (4) Multi-Objective: indicates whether the proposal uses a multi-objective search approach. The ✓ symbol is used to indicate that a category is supported. The symbol ∼ in the search category is used to indicate that the approach lacks any indication of which algorithms or methods are used.

<table>
<thead>
<tr>
<th>Name</th>
<th>Design</th>
<th>Run-Time</th>
<th>Search</th>
<th>Multi-Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jakoubi et al. (2009)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Churliov et al. (2006)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Asnar and Giorgini (2006)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fenz et al. (2009)</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Bai et al. (2007)</td>
<td>✓</td>
<td></td>
<td>∼</td>
<td></td>
</tr>
<tr>
<td>Conforti et al. (2012)</td>
<td>✓</td>
<td>MOA</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.14 – Comparison of risk treatment selection approaches in BPM.
diagrams is proposed in order to provide graphical annotations for the specification of security requirements within diagrams. This extension is defined by means of a UML profile called \texttt{BPSeq}. Their approach is very similar to that presented by \cite{Wolter2009}, although in this case the authors provide a set of security annotations to specify requirements in BPMN models. In \cite{Menzel2009}, an extension of business process models is defined that enables the generation of a security configuration of trust and data integrity with respect to a pre-determined level of security risk for an SOA environment.

There exist certain studies in the revised literature with respect to FODA and the generation of models. Certain efforts in the literature revised are focused on the generation of requirement models. In \cite{Semmak2010}, an extension of a goal-driven method (KAOS) is proposed to generate adaptive requirement models from variant models. In \cite{Mellado2010}, an approach which facilitates the development of secure software product lines (SPLs) and their derived products is proposed. In \cite{Perez2006}, feature models are used to analyse the variability requirements and consequently transform this feature model in order to generate an architectural model. Nevertheless, these approaches are focused on the generation of requirement models or of products related to software, while our approach strives to determine the configuration through feature models. In \cite{Sawyer2012}, the authors use feature-oriented domain analysis to provide self-adaptive systems by dynamically determining the best variants suited to specific QoS requirements. The authors use a goal model extended with attributes (characteristics of features) and soft-goals (QoS requirements) in order to represent the feature model, and by means of feature-oriented domain analysis, they determine which configuration is suitable for each context; thus values for goals and soft-goals are required. The main drawback in these approaches is that they use qualitative domains for attributes in order to determine variants as a consequence of using logic programming. In contrast, quantitative domains for attributes are supported in our approach.

5.7 Summary

In this chapter, the issue of the automatic and optimal selection of configurations of countermeasures for current BPMSs has been tackled. The main obstacles in this system are related to the lack of awareness in the IT security risk of business process-driven products. In the majority of cases, security is considered only as an afterthought. Selection and configuration of security countermeasures present a big challenge in BPMSs for several reasons: (1) it is a human, manual, time-consuming, and error-prone task; (2) it involves many security stakeholders, such as business analysts, security managers, and administrators; (3) the selection of configurations according to multi-criteria requires very high expertise. We conclude that it is necessary to provide business and security stakeholders involved in the development of business process solutions with tools that enable the automatic selection of security configurations in accordance with the needs of the organization.

A large proportion of the literature revised herein proposes an integration of risk information by extending graphical business process models in order to aid in the documentation of business process risk assessments. Nevertheless, these approaches fail to
take into consideration the integration of any mechanism to determine potential countermeasures. Regarding the generation of configurations, there are various authors who have enabled the specification of security in business process models although they focused on the authorization in a particular environment. Several studies apply feature-oriented domain analysis for the generation of secure feature models. On the other hand, there exist approaches that determine configurations in accordance with certain requirements in the quality of service. These approaches, however, use only a qualitative approach.

In this chapter, we propose the automation of the selection of configurations by means of feature models and constraint programming techniques. To this end, we first provide a catalogue of security patterns that model certain IT security countermeasures. Security patterns have been specified by means of the definition of feature models that enable selection and generation of configurations to achieve security goals of confidentiality, integrity, authentication and authorization, and availability. These feature models ascertain which characteristics must support the BPMS in order to achieve security objectives. In order to automate the generation of configurations, feature-oriented domain analysis mechanisms have been used. Feature models have been implemented through constraint programs that enable information about configurations to be inferred. Constraint programming techniques based on exhaustive and local searches improve the inference of configurations.

This analysis enables us to ascertain, for example, how many configurations are possible, and which configuration is the best option according to attributes and functions included in the model; it is even possible to ascertain whether a configuration is valid or not. Furthermore, these mechanisms provide a flexible and agile selection of configurations due to the easy-to-change of feature models by adding new attributes and functions. These attributes and functions enable the selection of configurations in accordance with multiple objectives. We have improved the selection of configuration by providing new search criteria based on a single objective, and on multiple and weighted objectives.

There exist various studies into the selection of controls based on multi-objectives, however these studies in some cases only indicate the use of algorithms with a lack of performance results. On the other hand, other work provides results but focus on the mathematical approach which disregards analysis by tools and configurations of countermeasures in a real BPMS. Our approach provides a good trade-off between the technical analysis of real systems, formalization, and provisioning of automatic techniques for the selection of configurations with regard to multi-objective techniques.
Chapter 6

Dependability for Executable Business Processes

I do not agree with what you have to say, but I’ll defend to the death your right to say it.
François-Marie Arouet (Voltaire)

6.1 Context and Motivation

In a previous chapter, it was shown how our contributions enable the assessment of risks in business process models and the inference of risk treatments as IT security configurations for those business processes. Business processes and countermeasures are implemented, enacted and integrated in a heterogeneous environment that communicates with third-party services, and with human participants, as shown in Figure 1.1. This implies non-deterministic and un-controlled points of faults during the execution of business processes, since certain resources remain outside the jurisdiction of the organization. Executable business processes are therefore non-free of faults in spite of risk treatments, due to the existence of un-controlled threats (i.e. zero-day threats) that must be taken into consideration. The materialization of threats can produce faults in the correct execution of business processes. It is therefore crucial to provide business processes with fault tolerance mechanisms to ensure the correct execution of a business process in spite of faults, and to prevent unexpected threats.

As stated in Chapter 2.6, fault tolerance aims to ensure the correct execution of systems by means of implementing error detection and recovery mechanisms. This chapter presents a contribution for supporting fault tolerance in executable business processes. Our contribution provides the complete development of a fault tolerance solution by means of a fault tolerance layer composed of two components: error detection and recovery. Most of the revised literature (cf. Related work 6.4) focused on presenting recovery mechanisms, by assuming previous error detection. A business process is composed of multiple parties such as tasks, participants and services that are orchestrated by a control-flow. Although
a fault in a element (task, service, participant, data, etc.) could influence other elements within a business process, generally, error detection is applied only on an isolated part of the business process. Due to these influences, error detection must be extended and analysed throughout the entire process. This complete analysis must detect and identify specifically which part of the process is failing.

In the following section, the fault tolerance layer for executable business processes and its components are described in detail.

### 6.2 Fault Tolerance Layer for Executable Business Processes

We propose the enhancement of BPM infrastructures (cf. Figure 1.1) by means of adding a Fault Tolerance Layer (FTL), such as that shown in Figure 6.1. The FTL is located between Enterprise Application Integration and the BPMS layer, and aims to determine and remedy faults on execution of business processes. To this end, FTL has been equipped with the following components:

1. The error detection component checks the behaviour and results of business processes by means of a fault detection stage. Business process outputs are compared with expected results obtained by an oracle (cf. Business Rules Engine in Figure 6.1). In the case of the detection of a discrepancy, there exists a fault. In the case of fault, the error detection component has been enhanced with a diagnoser component. This diagnoser performs a model-based diagnosis process (cf. model-based diagnosis in Figure 6.1) to identify which faulty components within the business process are failing.

2. The recovery component delivers a solution in order to correct the behaviour of business processes at run-time. The FTL has been provided with various recovery mechanisms based on three separate techniques:
   - Replication and redundancy, which deals with the concept of replication and dynamic binding of services.
   - Check-pointing and roll-back, which deals with sensors and rollback mechanisms.
   - Diversity, which deals with the development of n-version components.

Our approach deals with the following faults according to the taxonomy proposed by Chan et al. (2009): (1) Composition faults, which refers to faults produced in building services as a composition of other services; (2) Binding faults, which refer to faults produced during the binding stage of delivery and consumption of services; and (3) Execution faults, which refer to faults due to an unexpected result.

In the following subsections, error detection (cf. Section 6.2.1) and recovery components (cf. Section 6.2.2) are given in detail.
Figure 6.1 – Fault tolerance architecture perspective.
6.2.1 Error detection component

Error detection detects and identifies faulty components within business processes that produce unexpected behaviour and erroneous results. The error detection component is composed of an oracle in order to detect discrepancies and a diagnoser that deals with the identification of the faulty components.

The internal operation of the error detection component is described as a workflow diagram in Figure 6.2. This figure shows an error detection subprocess split into two main steps:

1. **Determine discrepancies** which compares the results of the business process against an oracle. In the case of any discrepancy, it is therefore mandatory to identify services that are involved in the faults by means of a model-based fault diagnosis.

2. **Model-based fault diagnosis** task performs an analysis of the instance of a business process in order to identify which service is involved in the error.

### 6.2.1.1 Determine discrepancies

The error detection process receives a business process model and business data inputs and outputs in order to perform a comparison of business process output data against an oracle. The business process model and data inputs and outputs correspond with the system model (SM) and observations (OBS) as defined in Definition 2.3.3 in DX community.
An *oracle* is a software-testing mechanism which determines whether a program works properly against a test (Baresi and Young, 2001). In general, an oracle provides comparison and evaluation capabilities. Furthermore, an oracle must determine whether the comparison of results is sufficiently close to be accepted.

We have implemented an oracle that enables the obtained results of business processes to be compared with data outputs. In the FTL, the oracle has been developed using a Business Rule Engine. Business rules must be defined using constraints similar to introduced in Section [4.2](#).

The oracle must store a model of the behaviour of business processes. That is, a business process model (SM) and the behaviour of services (cf. COMPS in Definition [2.3.3](#)) used within business processes must implemented using business rules. According to DX community, the oracle represents the system description (SD) such as defined in Definition [2.3.3](#) (cf. Chapter [2.3](#)). The business rule engine receives data inputs and determines the expected outputs that are then compared with business process data outputs (OBS). In the case of any discrepancy between output data (OBS) and oracle results, an inconsistency is detected, and a model-based fault diagnosis must be performed.

### 6.2.1.2 Model-based fault diagnosis

The *FTL* has been provided with a diagnoser (by means of a constraint solver) based on the model-based diagnosis. Model-based fault diagnosis has been used in business processes to detect and isolate faulty components (Borrego, 2012). In the *FTL*, a diagnoser has been implemented by means of Constraint Programming techniques.

The diagnosis is based on reified constraints and Max-CSP. As mentioned in Chapter [2](#) a reified constraint consists of a constraint and a variable which denotes its true value. An example of a reified constraint is shown in listing [6.1](#). In that case, C1 is the Boolean variable that denotes the true value and \( S1_{output} = a + b \) is the constraint to be satisfied. When a constraint is satisfied with any data then the Boolean value is true, and false otherwise. There exists a fault when a reified constraint is not satisfied.

\[
C1 == \{ S1_{output} == a + b \} 
\] (6.1)

A Max-CSP is a Constraint Optimization Problem (COP), which is a CSP which containing an objective function. This objective function is to satisfy the maximum number of reified constraints possible. Therefore, by modelling a business process and its services, we can determine which service is failing, using a Max-CSP.

The business process model and data-flow are thereby mapped into a Max-CSP. This proposal takes into account the relation between each activity within the business processes, its data (inputs and outputs), and the control-flow. Each activity is transformed into a reified constraint, for instance, service \( S1 \) represents an adder with two inputs (e.g. input \( a \) and input \( b \)) and \( M1 \) represents a multiplier with two inputs (e.g. input \( c \), and \( d \)), and input \( d \) of \( M1 \) depends on result of \( S1 \). These conditions are defined using reified constraints, such as that shown in statement [6.2](#). The reified constraint \( C2 \) is a constraint that reflects the relation between the output of \( S1 \) and to input of \( M1 \).
\[
C_1 == \{S_1_{output} == a + b\} \quad (6.2)
\]
\[
C_2 == \{d == S_1_{output}\} \quad (6.3)
\]
\[
C_3 == \{M_1_{output} == c \ast d\} \quad (6.4)
\]

Objective : \textsc{MAXIMIZE}(C_1, C_2, C_3) \quad (6.5)

The objective (cf. in the listing above MAXIMIZE(C1,C2,C3)) is to determine which components are involved in the errors by means of identifying which constraints are unsatisfiable. To this end, Max-CSP is fed with data inputs from the business process. The Max-CSP objective function aims to maximize the number of activities which are not responsible for the unsatisfactibility of the CSP. That is, the objective function maximizes the number of Boolean variables instantiated to true. For instance, that \(C_2\) is unsatisfiable implies that an error occurs in \(d\) and/or \(S_1_{output}\). Once this maximization is performed, it is necessary to ascertain the activities responsible for the inconsistency. To this end, we take the sets of activities related to the constraints whose associated Boolean variable has been instantiated to false, and find out the minimal collection of activities which cover all those sets. According to DX theory, the best way to find that activity or activities is by calculating the minimal hitting sets (such as shown in Chapter 2.3), which provide us with the minimal diagnosis for a business process.

### 6.2.2 Recovery component

As mentioned earlier, various modelling languages exists for business processes. Nevertheless, these languages are mainly destined to design business process models. Most of these language provide a notation with a palette of elements to define business processes from a control-flow perspective. There are other more sophisticated languages, such as BPMN, which is a standard language created by OMG ([Object Management Group (OMG)](http://www.omg.org)), that provides a long list of elements that facilitate the management of events, such as intermediate events to catch and throw errors, events of interchange of messages, events to signal across different processes. Nevertheless, BPMN is an unexecutable language. There is a section in BPMN specification which describes a transformation of BPMN models to Business Process Executable Language (BPEL) ([BPE](http://www.bpel.org)). BPEL is considered as de-facto standard for the implementation of executable business processes. BPEL is an XML language for the definition of business processes as a composition of Web Services. BPEL provides various programming structures, such as if-then statements, event handling, try-catch statements, and throw errors.

BPEL processes define a business process as a composition (orchestration) of Web services. Web services (of either external or internal organization) are linked at design time within a business process. However, Web services within BPEL instances cannot be modified at run-time. Assuming no design faults exist in the processes, if a fault occurs during execution due to unexpected results, for instance, a change of functionality, or a mismatch of parameters, then it is impossible to replace the faulty service during run-time once a business process has been deployed. Faults could therefore be solved
by: (1) stopping for every instance of a business process; (2) locating the faulty services (diagnosis); and (3) replacing the faulty service with another correct service. Our approach strives to perform this process in an automatic way without stopping business process execution, through by replacing of faulty services at run-time.

BPEL is able to implement fault-tolerance business processes, such as those shown in (Dobson 2006). In the following subsection, we describe the fault-tolerant patterns implemented as part of the recovery component of the FTL.

### 6.2.2.1 Replication and redundancy pattern

The replication and redundancy pattern is based on two aspects: (1) dynamic binding of services; and (2) primary/backup replication. That is, dynamic binding is a technique that allows the binding of services at run-time (Erradi and Maheshwari 2008; Küster and König-Ries 2006). This pattern applies the concept of redundancy, in the sense of replication of services. This approximation enables business processes to have one service as its principal (primary), and one or more replicas as backups (backups). This solution uses a passive replication approach, as defined in Chapter 2.6.

In the case of faults, it is possible to change the execution of business processes binding a backup service instead of the primary service. To this end, a binder component is introduced between business processes and services acting as proxies, as shown in Figure 6.3. The Binder component is communicated with error detection component, services layer and business processes. The binder component determines, based on the error detection information, what services to invoke: either the primary service (indicated as coloured arrows), or backup services (indicated as dashed coloured arrows).

This solution is fault tolerant since, in the case of detection (in the error detection component), every faulty service invocation can be replaced with an invocation to a backup service. This is a good solution although certain deficiencies are presented, such as the introduction of a unique point of fault at the binder component, which in turn could introduce a very high overhead in the performance of a business process. In order to solve this problem, the replication of the binder can be applied, thereby also, replicating (cf. passive replication) the binder component (one primary and several backup replicas), as shown in Figure 6.4. In this case, binder backups are defined as passive replication. Therefore the backup binder takes control iff the binder faults, thereby enabling the execution to continue. The binder backup can be an exact copy of the original but located in different external server or machine to that of the primary copy. BPEL provides various mechanisms, such as fault and compensation handlers, to achieve the implementation of primary-backup binder solutions (Dobson 2006).

### 6.2.2.2 Diversity and N-Version Programming pattern

As introduced in Chapter 2.6, software cannot be degraded in the same way as physical systems. If a fault is detected in a software component, it is due to an implementation fault (bug), and which cannot be solved by means of redundancy. Diversity is a very important factor in obtaining dependable software systems (Pullum 2001). The main goal of diversity is to provide identical services (variants) but with a separate design and
Figure 6.3 – Fault tolerance pattern with binder.

Figure 6.4 – Fault tolerance pattern with replicated binder.
6.2. FAULT TOLERANCE LAYER

implementation in order to minimize provoking identical faults. There are various techniques for fault-tolerance software based on multi-version (diversity of software), Pullum (2001); Torres-Pomales (2000): N-Version Programming (NVP), Recovery Blocks, N-Self Checking Programming. The N-Version Programming paradigm considers the utilization of diversity for implementation and design components. Thus, every service used in the implementation of a business process will be developed as an N-Version Component, as shown in Figure [2.8].

In our approach, a solution based on N-Version Programming is adopted, which is a static technique where an activity is executed by various processes or programs and the result is only accepted by a majority of votes. This mechanism for obtaining results is defined in NVP as an adjudicator or decision mechanism (DM). Several DMs are defined in Pullum (2001). However, we use an approach based on the consensus of votes.

The implementation selected for the N-Version components follows the basic ideas of N-Version Programming. An N-Version component provides at most $2X + 1$ replicas, where $X$ ranges from 1 to $n$, and where $n$ is an integer greater than 1. Components have been developed with an adjudicator (DM) in order to obtain the output results, see Figure [6.5]. The logic within the adjudicator (DM) can range from very basic to seriously complicated. NVP components can be improved by adding new features, for example, by developing new strategies in the adjudicator. However, the more complexity added into the adjudicator component, the more overhead is introduced into the execution of the N-Version components.

By using N-Version components, not only is fault-tolerance achieved, it also supposes another advantage since the error detection stage is rendered totally unnecessary. For example, if one of the variants fails to return a response in time, the adjudicator takes the results from the other variants. In consequence, error detection can be eliminated by using this mechanism, although it could result in a very high cost in development and performance. On the other hand, there exists a unique point of fault in the adjudicator component, although a backup solution, similar to that of the binder, can be adopted.

6.2.2.3 Check-pointing and Roll-Back pattern

The check-point approach is based on the idea of saving the state of the business process, and, in the case of fault detection, recovering the execution of the system from the check-point where the state was saved (cf. Figure [6.6]). We propose the simulation of a check-point approach in services, whereby a recovery mechanism is launched only in the case of faults. The fault-tolerance approach mechanism is composed of two parts:

- **Sensors (Check-points).** An integrity sensor is modelled as a Max-CSP. Sensors receive data information about data inputs and outputs from the services, with which the CSP is then defined. Max-CSP resolutions help to identify and isolate the services which are failing in run-time.

- **Compensation handlers (Roll-back).** These are specific elements of business processes which allow the limitation of the effects created by a process when faults occur. Compensation handlers allow the process execution to be rolled back from
Figure 6.5 – Fault tolerance pattern with N-Version components.
a specific point, thereby executing a set of tasks to undo the transactions already initiated. Compensation handlers are explained in the following section.

Check-point approaches present some drawbacks in comparison with other approaches: they require the introduction of sensors into the business process design, they need extra time to check each sensor, and also for the recovery of business process services in roll-back. In fact, the correct and minimal localization of sensors inside a business process is a highly complex task, [Borrego et al. (2010a)]. We assume that sensors are already located within the model.

For high availability, the check-pointing solution is not suitable, since for long business processes, the roll-back mechanism could introduce a very high overhead in the case of a fault. However, if very high availability in our business processes is needed, then a solution with a binder backup is the best solution. Nevertheless, if very high availability is not needed, then a binder approach alone could be sufficient. If the business process with a very high level of correctness in outputs is needed, then NVP is the best solution, although an evaluation of the number of replicas needed must be carried out, since with a specific number of replicas, then a very high level of correctness could be ensured despite the introduction of a very high load on the development and in the adjudicator logic.

In next section, an illustrative example is given in conjunction with a comparative study of the performance of each fault-tolerance pattern.
6.3 Illustrative example

In order to clarify the proposed fault-tolerance alternatives, an illustrative example is developed. The example in Figure 6.7 is an adaptation of the 'PolyBox Problem' used by Cordier et al. [2000] to illustrate classic diagnosis problems. In this process, there is a set of services, \( S = \{S_1, S_2, S_3, S_4, S_5, M_1, M_2, M_3\} \), a set of inputs, \( I = \{a, b, c, d, e, f\} \), and a set of outputs, \( O = \{g, h, i\} \). The system is made up of three services \( (M_1, M_2, M_3) \) with the same functionality \( M(x) \) but they are independent; and \( (S_1, S_2, S_3, S_4, S_5) \) are another five services with the same functionality \( S(x) \) but also independent.

The process has been distributed in three separated business processes (BP1, BP2, and BP3) which are deployed in three different BPMS, BPMS1, BPMS2, and BPMS3. BP1 invokes the services \( S_1, S_2 \) and \( S_3 \). BP2 invokes the services \( M_1, M_2 \) and \( M_3 \) and relies on BP1. BP3 invokes the services \( S_4 \) and \( S_5 \), and relies on BP2 and BP1. In the scenario, it is possible to highlight numerous aspects. For instance, a fault within service...
6.3. ILLUSTRATIVE EXAMPLE

SI has a direct effect on the BP1 results, and, as a consequence, BP2 is also affected, and finally BP3 will be affected, and hence the final result of the complete process will be not correct. However, with the correct fault tolerance mechanism this will not be the case.

In order to study the behaviour of the fault tolerance layer, a set of test cases has been executed for each case of the fault-tolerant approach, as described in previous sections.

6.3.1 Performance Results

The tests developed have been separated into two groups: the first considers only one thread of execution, and the second group considers more than one thread of execution. For each thread, a number of invocations 300, 1000, 1500 and 2000 are carried out. The tests simulate the idea of there is an integrity fault in a single component for each request, for instance SI. In the solution with a replicated binder a fault in the binder has also been simulated. Regarding data inputs, a set of random value tests has been created for each test case.

Although many parameters could be useful in order to measure dependability properties, in our proposal, the most interesting in the comparison is the performance time. Performance gives a clear indication of the difference between one solution and another to be measured in terms of delivery. Other metrics can be derived from this data. For instance, the MTTR metric introduced in Chapter 6. MTTR can easily be determined by subtracting the time of a solution without fault tolerance from that of a solution with the fault-tolerance approach.

The test results have been obtained in two tables as shown in Figure 6.8. The tables describe performance results of considering without fault tolerance and the recovery methods used. The figure shows two graphics which contain results on average performance time. The hardware used in the execution of the tests is: various servers Intel Xeon E5530 2.4 GHz, with 8GB RAM and a Debian Gnu/Linux 64bits OS; and a client Intel Core 2 Duo T9300 2.5GHz with 4 GB RAM.

Discussion of results

Regarding the performance results (cf. Figure 6.8), we can observe that all solutions introduce an overhead with respect to the example without fault tolerance (cf. MTTR in previous chapter) due to the extra computation required for error detection and deployment of recovery. Nevertheless, there are two key concerns to take into consideration to understand these results: (1) overhead due to design aspects, and (2) overhead due to extra computations. Below, the recovery patterns are compared with regard to these concerns.

Overhead of extra computations

The binder solution obtains the best performance results. However, the performance may be distorted by the overhead introduced for extra computation of the diagnosis process within the error-detection component. In order to skip the diagnosis process, we could
Figure 6.8 – I. Performance with one thread for execution. II. Performance with more than one thread for execution.
6.4 RELATED WORK

pre-compile the diagnosis of the business process that uses fault signatures, by using an FDI approach such as that introduced in Section 2.3.2. In the case of complex business processes, the diagnosis process may require a longer time of analysis. Therefore, the binder solution could be distorted by high overhead due to the extra computation of the diagnosis process. Furthermore, this overhead could be increased in the case of a binder with backup. This fact can be observed when comparing results with regard to NVP. The NVP approach obtains very similar results to those of the binder. However, it is only in the case of one thread of execution where NVP obtains better results since NVP avoids the diagnosis process. A vast number of variants (executions for each variant) could worsen the complexity of the performance of the adjudicator. We can observe how the increasing of threads of execution worsens the NVP behaviour due to the extra computations carried out in the adjudicator and the need to execute multiple variants. Moreover, the checkpointing approach achieves the worst results due to the high overhead introduced by the analysis performed in sensors and by the need to execute compensation handlers. In the worst case (cf. Figure 6.8 - I), the overhead introduced by check-pointing is almost double that of the NVP approach.

Overhead of design

Regarding the binder, most of the overhead of design is focused on the binder component and extra logic introduced in the case of using a backup for the binder. In the development sense of a NVP approach, we must balance the time spent on developing variants. NVP can be compared against a solution with binder using only a primary-backup solution. In the case of multiple faults, the binder solution may be insufficient for the replication (primary and backup) of services to ensure the correction of services. NVP components along with the correct number of replicas can provide better results with a very high level of correctness. Nevertheless, we should measure the trade-off in any case. Moreover, check-pointing approach requires a higher overhead of design in locating sensors plus extra work designing compensation handlers within business processes. As mentioned earlier, the location of sensors could be a complex task (Borrego, 2012). Therefore, a very high overhead of design could be introduced in the case of large and complex business processes.

6.4 Related work

There exist only few approaches in the area of business processes, and fault tolerance. In (Baresi et al., 2006), the authors propose a framework entitled Dynamo that strives to ensure dependability in business processes. This framework provides a run-time business process supervisor that guarantees that the requirements of dependability are satisfied. The main contribution is focused on the supervision of business processes by means of the definition of two languages, WSCoL and WSRS. WSCoL is employed to define monitoring expressions, and WSRS is employed to define remedial (recovery) strategies to be carried out when a monitoring expression is activated. WSRS defines ten different atomic remedial actions: ignore, notify, halt, retry, rebind, monitoring rules, change process params, call, and process callback. These strategies can be combined in order to define a more
complex remedy. Regarding a fault tolerance perspective, the monitoring expressions act as error detection mechanism focused on detecting a violation of the condition of a business rule. Although this approach is general, there exist certain dependability properties, such as integrity, availability, and confidentiality, that cannot guarantee the use of this approach as the authors stated in the conclusions.

In the scope of fault tolerance for service-based executable business processes (BPEL) and Web Service composition, there are many contributions [Dobson, 2006; Huang et al., 2006; Modafferi et al., 2006; Wang et al., 2009]. In [Dobson, 2006] and [Onditi et al., 2008], Dobson et al. study the feasibility of the BPEL language to implement fault-tolerance patterns. BPEL provides constructors for fault handling (event handlers, throw errors, and catch events) and error handling (compensation handlers). Nevertheless, the authors focused on presenting a tool to address the design of fault tolerance in the BPEL process, despite the absence of an infrastructure for the error detection and recovery mechanisms; as explained in Chapter 2.6. In [Wang et al., 2009], the authors propose different remedial strategies related to a proposed fault taxonomy. This taxonomy classifies faults into four categories, referring to four different contexts: (1) functional, (2) quality of service, (3) domain, and (4) platform. The proposal strives to present a set of remedies for each type of fault. Error detection consists of the violation of restrictions expressed as pre and post conditions as an extension of BPEL languages. The proposal is presented as a framework structured in various components. The most relevant components are: composition, analysis and instrumentation. The composition forms services and business goals; the analysis translates business processes with a remedial strategy using remedial databases; and the last component, that is instrumentation, translates the business processes into a final BPEL process. The authors present a taxonomy of faults to be handled: however, error detection is overlooked since it is indicated as an analysis process.

An architecture for self-healing BPEL processes is presented in [Modafferi et al., 2006]. Self-healing properties imply the development of many elements integrated into the same engine, such as a monitor, a diagnoser, a planner of changes, and a validation mechanism. Certain recovery strategies have been adopted in this work and integrated into the engine. In [Modafferi and Conforti, 2006], an extension of BPEL language to implement five recovery mechanisms is presented. Although, this work is focused on the design of recovery mechanisms, no error detection mechanisms are taken into consideration in this proposal.

There are more general initiatives into introduction of fault-tolerance techniques in the area of Web Service, [Fang et al., 2007; Luo and Yang, 2002]. In these studies, the main contributions are the definition of a framework or middleware to achieve fault-tolerant service platforms. A fault-tolerance architecture for SOAP protocol is proposed and the solution is compared against the flexibility of CORBA solutions, [Fang et al., 2007]. In [Luo and Yang, 2002], the authors have defined and developed a mechanism to improve resilience to faults in Web service clusters to enhance the reliability of the services. Another study is focused on the dynamic selection of Web Services for the construction of optimal workflows, [Huang et al., 2006]. The selection of optimal services are based on the search for services from various repositories and data stored in databases. Although this technique appears to provide a fault tolerance solution, this is solely due to the workflows being built instantaneously through the selection of the best service each.
6.5. SUMMARY

The majority of fault-tolerance solutions are based on replication and recovery techniques. Although replication is a very important concept in fault-tolerant systems, when it is necessary to create fault-tolerance in software, the solution of replication is insufficient. The philosophy in fault tolerance software is totally different; the techniques are based mainly on software diversity and data diversity, (Pullum, 2001) (Torres-Pomales, 2000).

On the other hand, in fault tolerance of distributed systems, checkpointing and rollback recovery approaches are popular (Baldoni, 1997; Cao and Singhal, 2003; Kim and Park, 1993). Further proposals have been developed in other domains, such as grid computing (Shi et al., 2010) and Web Services (Baresi et al., 2006).

A comparative study of the most relevant approaches is shown in Table 6.1. This comparison is carried out in accordance with the following categories: (1) Language: indicates the business process language used in the approach; (2) Recovery: indicates whether the approach uses or defines any recovery mechanism; (3) Detection: indicates whether approach presents or defines any error-detection mechanism; (4) Diagnosis: indicates whether the approach applies any diagnosis technique; (4) Design: indicates whether the approach is oriented towards application at the design stage of business processes; (5) Runtime: indicates whether the approach is oriented towards application at run-time stage of business processes.

An innovative fault tolerance approach (Fault Tolerance Layer, FTL) is proposed in order to provide dependable capabilities for executable business processes. FTL has been defined in order to be used in any business process infrastructure located between the BPMS and Enterprise Application Integration. That is, FTL is located between business processes and services.

The proposed framework attains the provision of a fault-tolerance layer composed of two main components: an error-detection component, and a recovery component. On the one hand, error detection provides a mechanism for the detection and the identification of errors during the execution of business processes. On the other hand, the recovery component provides mechanisms that enable faulty components to be replaced or avoided. That is, error detection detects and identifies the faulty services within the business process and communicates to recovery components which services must be replaced.

In our approach, error detection is composed of two main parts: a Business Rule Engine as an oracle that enables results of the business processes to be compared in order to detect discrepancies between business process data outputs and expected results; and a Constraint solver as a diagnoser that uses model-based diagnosis theory and constraint programming techniques to automate the identification of faulty services within business processes at run-time. In the revised literature, there are only a few contributions that provide a complete approach to error detection and recovery mechanisms at run-time, as presented in our approach. Most of the approaches lack the presentation of effective error detection and none of the revised approaches includes a diagnosis of business process in
### Table 6.1 – Comparison of fault-tolerance approaches for business processes.

<table>
<thead>
<tr>
<th>Name</th>
<th>Language</th>
<th>Recovery</th>
<th>Detection</th>
<th>Diagnosis</th>
<th>Design</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dobson [2006]</td>
<td>BPEL</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modafferi and Conforti [2006]</td>
<td>BPEL</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Baresi et al. [2006]</td>
<td>BPEL</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Onditi et al. [2008]</td>
<td>BPEL</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wang et al. [2009]</td>
<td>BPEL</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>FTL</td>
<td>BPEL</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
order to identify faulty components.

In the sense of fault tolerance, FTL integrates various fault-tolerance patterns developed on BPEL: the first solution creates fault-tolerant executable business processes using dynamic binding techniques and replication of services; the second solution presents an improvement introducing a redundant binder; a third solution uses the concept of software tolerance and implements NVP components; and a fourth solution provides a simulation of a check-point approach. Most of the revised literature focuses on studying the feasibility of business process languages for the implementation of fault-tolerance approaches. Nevertheless, none of these approaches provides an integrated approach which enables selection between multiple fault-tolerance patterns. In order to compare the solutions, a case study along with a performance comparative is carried out for each solution. We have concluded how the MTTR introduced by the FTL proposal is permissible in the worst-case scenario with regard to solutions without fault tolerance. Moreover, we have discussed the pros and cons of the overheads of developing and using each approach.
Part IV

Conclusions and Future Work
Chapter 7

Final Remarks

Your work is going to fill a large part of your life, and the only way to be truly satisfied is to do what you believe is great work. Don’t settle.
Steve Jobs

The current Thesis Dissertation presents three main contributions to improve the dependability of the life-cycle of business process management in both design and run-time stages. These contributions have been orchestrated under the umbrella of the OPBUS framework. The framework presents an enhancement of the business process management life-cycle with risk management capabilities at design time and fault tolerance at run-time.

At design time, we provide risk-aware business process modelling with automatic techniques that enable a risk assessment to be performed. Other studies in the revised literature provide extensions to the business process model in order to aid the documentation of the risk assessment; however, they overlook the provisioning of mechanisms for the risk estimation, and the risk evaluation based on those extensions. We propose a risk model extension of business process models which provides elements for the risk identification and the establishment of business objectives, such as a risk criteria of the acceptable level of risk. Moreover, we provide automatic mechanisms for risk estimation and risk evaluation that enable non-conformance to be identified with regard to the risk criteria established in the business objectives:

• Chapter [1] presents a light extension (risk model) of business process models that enables the risk identification and the establishment of business objectives. Thus, the risk model indicates in a separate threat scenario attached to business processes, which vulnerabilities and threats affect the elements of the model. A threat scenario presents a casuistic of vulnerabilities and threats that should be evaluated for the business processes. Vulnerabilities are then linked to the elements of the model (activities and artifacts).

This extension has been defined to be as general as possible in order to be adapted to any risk assessment methodology. To this end, the elements of the model can be
estimated with regard to various security dimensions (the dimensions considered in the extension or new dimensions), and the metrics and scales reflect the uncertainty by means of the use of a quantitative range of values instead of single values. The range of values increases the complexity when risk estimation is carried out due to the combination of values of the various elements involved in the computation risk.

Risk assessment strives to compute the risks (risk estimation) in order to evaluate (risk evaluation) whether risks are acceptable in accordance with business objectives. Our contribution provides automatic techniques that perform risk estimation and risk evaluation of an entire business process. Risk estimation presents a challenge since there are currently no specific risk estimation methods for the estimation of the risk of a business process model. We propose computing risk estimation based on various control-flow patterns. These patterns describe how to compute the risk of an entire business process following the control-flow and vulnerabilities attached to the elements of the model. Thereafter, risk estimation and business processes are diagnosed in order to identify which elements within the model are non-conformance to the expected risk criteria.

In order to automate risk estimation and the risk evaluation we provide a verification mechanism that enables the model-based diagnosis and constraint programming techniques to automate it. Thus, business process models and their extensions are transformed into constraint programs that are evaluated using the constraint solver. In order to generalize the proposal for adaptation to any risk assessment methodology, the transformation is customized with information relative to the risk assessment method, constraint solver and type of search to be used. The constraint solver evaluates the model by providing a result of non-conformance or conformance of the business process. If there is any non-conformance, then the elements of the business process model involved in the non-conformance are identified.

These contributions have been addressed in three separate publications (Varela-Vaca et al., 2011a, 2010b, 2012). Furthermore, a tool has been developed (Varela-Vaca, 2012) (Varela-Vaca et al., 2011c) to support the risk analysis, and the automatic risk estimation and risk evaluation of the business process models. In addition, an article has been submitted (currently in second revision) to a indexed journal (Varela-Vaca et al., 2012).

Once the non-conformance of risks are identified, a risk treatment must be defined. There is a vast list of IT controls that must be set up with a large number of configurations. In general, security controls are defined in a textual manual with a lack of formalism. The main problem involves determination of which configuration is valid to treat which vulnerability, while also fitting the business goals. Other studies in the revised literature strive to select controls relative to abstract aspects but overlook the IT configurations a final step. We provide a formalization of security controls for business process models based on security patterns and feature models. Moreover, we propose the use of FODA techniques in order to inference and reason in the selection and generation of IT security configurations as countermeasures for business processes:
• Chapter 5 describes a proposal for the formalization of security controls based on security patterns. Security patterns provide a common structure to align problems (threats and vulnerabilities in our case) and solutions (security configurations). Nevertheless, security patterns are often too textual and lack formalization. We provide an extension and formalization of security patterns based on CWE (CWE, 2011) and feature models. To this end, we propose using the identifiers of CWE to describe the problems, and a formalization based on feature models as a formal model to describe other parts of security patterns.

As a proof of concept, we have defined a catalogue of various security patterns to treat vulnerabilities that affect different security dimensions. These patterns have been equipped with feature models that define the combinatorics of configurations available to achieve certain security goals and intentions. These feature models are described using a constraint programming formalization that enables the inference and reasoning in the selection of configurations to achieve one or more objectives. Furthermore, these feature models are extended with metrics such as cost, risk reduction, etc. We propose using FODA techniques to infer information relative to the feature models. We have defined various operators based on FODA operations for their application in the feature models for the selection and generation of configurations that fit an objective. The objective may be to reduce the risk to the minimum, while taking into consideration the cost of the solution. In that case, a multi-objective search is applied over the pattern and the suitable solutions are obtained. We have carried out several performance studies in the selection of configurations with regard to one criterion and to various optimization criteria, with very good results in time and with an adequate number of solutions.

The proposal of security patterns as countermeasures has been addressed in (Varela-Vaca et al., 2013). Subsequently, the formalization of security patterns with feature models was published in (Varela-Vaca and Gasca, 2012). The catalogue and the performance study in the selection and generation of configurations have been accepted for publication to a indexed journal (Varela-Vaca and Gasca, 2013).

In run-time, we provide the execution of business processes with fault tolerance mechanisms that enable error detection by means of fault diagnosis techniques and the deployment of recovery mechanisms based on classic and software fault-tolerance solutions. Other studies in the literature focus on the provision of an error detection or on the description of recovery mechanisms using executable business process languages. Our contribution provides a combined solution using techniques that are not employed by other authors:

• Chapter 6 presents a contribution that provides business processes with fault tolerance capabilities at run-time. The contributions present a fault tolerance layer, that may be integrated into a BPMS, composed of two main parts: (1) an error detection mechanism based on business rules (oracle) and model-based diagnosis using constraint programming techniques for the detection and isolation of faulty components; (2) definition of various fault tolerance patterns based on replication
and redundancy, dynamic binding of services, check-pointing and roll-back, and diversity.

Error detection mechanisms compare the results of the business process with the results of the oracle. In the case of any discrepancy, the model-based diagnosis is applied in order to detect and isolate which component of the business process is responsible. Once faulty components are detected, error detection communicates the decision to recovery components to release a recovery mechanism. We have simulated a scenario with different business processes distributed in different BPMSs that interact, in order to measure the overhead (MTTR) of the recovery mechanism.

Recovery mechanisms have been described in (Varela-Vaca and Gasca, 2010; Varela-Vaca et al., 2010a). On the other hand, error detection and performance study have been published in (Varela-Vaca et al., 2011b). The paper published in (Varela-Vaca et al., 2010a) was awarded the category of best paper in the international conference of dependability DEPEND 2010.

In short, this Thesis Dissertation provides various quality improvements at design and run-time of business processes using diagnosis and constraint programming techniques. The contributions provide an enhancement of business processes by means of the improvement in the dependability at design with automatic techniques for the risk assessment, generation and selection of optimal risk treatments, and run-time with fault-tolerance mechanisms.
Chapter 8

Directions of Future Work

"... your future hasn’t been written yet. No one’s has. Your future is whatever you make it. So make it a good one.”
Dr. Brown. Back to the future III

The current Thesis Dissertation presents three main contributions that can be extended with new research topics to be addressed in future research:

8.1 Diagnosis of Non-Conformance of Risks in Business Process Models

Risk assessment of business process models has been tackled in Chapter 4. Our approach only considers aspects related to control-flow; however it would be interesting to generalize the proposal by means of the enhancement of the risk estimation of business process models, through considering other aspects, such as data-flow perspective, resource importance (Fenz et al., 2009) and probability of execution branches (Xue et al., 2012).

In general, risk assessment is addressed at design time (Suriadi et al., 2012) however it would be interesting to carry out our approach at run-time. That is, to estimate and evaluate risks during the execution of business processes or apply our approach in order to carry out a post-execution analysis.

8.2 Risk Treatment for Business Process Models

In Chapter 5 an approach for the selection and generation of configurations in an optimized way is given. We provide a formalization of security controls using security patterns and feature models. Nevertheless, the proposal only takes into consideration those controls which affect one dimension or various dimensions in accordance with the definition of the feature model.
We can extend security pattern formalization in order to reflect the dependency with regard to other controls. Dependency of controls can be explored by means of techniques, such as DEMATEL [Wen-Hsien et al., 2012] and AHP [Fang et al., 2007]. We can enrich notation of feature models with a new operator to represent dependency between feature models. Furthermore, it would be very interesting to extend searches to involve various feature models.

8.3 Dependability in execution of business

An approach to fault tolerance at run-time is given in Chapter 6; this approach is focused on low-level aspects. Thus, we focus on the presentation of the implementation of a fault-tolerance layer in systems. It would be interesting to provide the business process modelling languages with a palette of constructors that, at design time, enable the specification of where and which fault tolerance must be applied. For instance, BPMN can be extended with new events to specify fault tolerance in the boundary of the activities.

Among the disadvantages stated in the check-pointing approach is where and how to allocate the check-points. An analysis of check-point allocation could be carry out in order to minimize the number of check-points required to ensure a robust check-pointing solution. To this end, it could be advantageous interesting to adopt an approach such as that given in [Borrego et al., 2010a].

On the other hand, it would be interesting to introduce other diagnosis techniques that improve the diagnosability of errors during run-time, such as that described in [Borrego, 2012].

Another potential direction for our research could be to extend OPBUS to complete Edward Deming’s PDCA life-cycle by introducing a monitoring stage:

8.4 Risk Monitoring for Business Processes

Although risk treatments are deployed, it is crucial to state security metrics within the business process models as indicators to monitor the correct operation of the counter-measures. In general, metrics are commonly used by organizations as Key Performance Indicators (KPI) [González et al., 2011] to evaluate its success or the degree of success. These indicators must be periodically monitored and checked in order to detect possible deviations of countermeasures. The main research topics that could be addressed include:

- which KPIs have to be selected with regard to the countermeasures used.
- where KPIs have to be attached within a business process model and when KPIs have to be measured during the execution of the processes.
- how KPIs provide feedback for the risk assessment.

Therefore, it would be interesting to include a risk monitoring phase at the design time of business process models that enable to KPIs to be automatically extracted in
order to measure countermeasure effectiveness and attach them within business process models. Furthermore, it would be interesting to carry out a tracking and logging of KPIs during the execution of the business process (run-time) in order to provide feedback for the risk assessment and risk treatment phase. It would be very interesting to apply process mining techniques for the analysis of tracks and logs in order to retrieve information on the effectiveness and correctness of the controls.
Part V

Appendices
Appendix A

Transformation Template for Business Process Models

In Chapter 4, OPBUS tools has been presented in order to support the diagnosis of non-conformance of risks in business process models. This tool (Varela-Vaca, 2012) has been provided with transformations from business process model to a CSP in order to evaluate the risks such as described in Section 4.3.3. The transformation has been developed by Epsilon Transformation Language (ETL) (Kolovos, 2012). The ETL template below defines a transformation for obtaining a COMET constraint model for each pool within a business process model. There are other auxiliary functions used in the transformation that are not been attached in this appendix although they are available to download in (Varela-Vaca, 2012).

ETL transformation template

```%
var poolGraph : Map();
var poolStart : Map();
var poolEnd : Map();
var poolFormula : Map();
var mapaListaRiesgosPorTarea : Map();
var mapaListaTratamientosPorRiesgo : Map();
var mapaListaTratamientosPorRiesgos1 : Map();
for(p in Pool.allInstances){
  p.name.print();
  // Funcion tarea-riesgos
  for(x in Task.allInstances){
    if(p.vertex.indexOf(x)<-1){
      var listaRiesgos:Sequence();
      for(r in x.risk){
        listaRiesgos.add(r.name);
      }
      mapaListaRiesgosPorTarea.put(x.name,listaRiesgos);
    }
  }
  // Funcion riesgo-tratamientos
  var ts := AssessmentFlow.allInstances.select(e|e.source==p);
}[
```
for(tsi in ts){
    for(r in tsi.target.risks){
        var listaTratamientos:Sequence();
        for(s in RiskTreatmentFlow.allInstances){
            if(not listaTratamientos.includes(s.source.name) and s.target.equals(r)){
                listaTratamientos.add(s.source.name);
            }
        }
        mapaListaTratamientosPorRiesgo.put(r.name,listaTratamientos);
    }
}

// Creation of graphs
var graph: new Map();
var start: new Activity;
var end : new Activity;
var graphnames: Map();
var graphi: Map();
var formula : String;

// Creating the nodes
for(v in p.vertex){
    graph.put(v, new Sequence);
    graphnames.put(v.name, new Sequence);
    graphi.put(v, new Sequence);
    if(Start.allInstances.includes(v)){
        start = v;
    }
    if(End.allInstances.includes(v)){
        end = v;
    }
}

// Creating the edges
for(s in p.sequenceflow){
    if(p.vertex.indexOf(s.source)<>-1 and p.vertex.indexOf(s.target)<>-1){
        if(not graph.get(s.source).includes(s.target)) {
            var h := graph.get(s.source);
            var h1 := graphnames.get(s.source.name);
            graphnames.get(s.source.name);
            h.add(s.target);
            h1.add(s.target.name);
            graph.put(s.source,h);
            graphnames.put(s.source.name, h1);
        }
        // Si queremos considerar bidireccional el grafo
        var h := graphi.get(s.target);
        h.add(s.source);
        graphi.put(s.target,h);
    }
}

poolGraph.put(p, graph);
poolStart.put(p, start);
poolEnd.put(p, end);

// Calculate pool-formula
formula = p.getFormula(start,graph);
poolFormula.put(p, formula);
var prueba := poolGraph.entrySet();
for (p in Pool.allInstances) {
    var totalvariables:Integer=0;
    var listaVariables:Sequence();
    var listaVariablesAsset:Sequence();
    var listaVariablesAssetAlternative:Sequence();
    var listaVariablesRisk:Sequence();
    var listaVariablesTreatment:Sequence();
    // POOLNAME: [%=p.name%]
    [% if(searchType.toUpperCase().equals("LOCAL")) { %]
        import cotls;
    [% } else if(searchType.toUpperCase().equals("LINEAR")) { %]
        import cotls;
    [% } else { %]
        import cotfd;
    [% } %]
    [% if(searchType.toUpperCase().equals("LOCAL")) { %]
        Solver<LS> manager();
        ConstraintSystem<LS> system(manager);
    [% } else if(searchType.toUpperCase().equals("LINEAR")) { %]
        // Otra cosa ...
    [% } else { %]
        Solver<CP> manager();
    [% } %]
    int t0 = System.getCPUTime();
    // Acceptable risk level
    int Acceptable_Risk = [%=p.Acceptable_Risk%];
    // Tasks variables
    [% for (i in p.vertex) { %]
        if(not Start.allInstances.includes(i) and
           not End.allInstances.includes(i) and
           not Gateway.allInstances.includes(i)) {
            totalvariables = totalvariables+5;%
            range [%=i.name%]_value_integrity = [%=i.valueIntegrity.substring(1,2) %]..
                          [%=i.valueIntegrity.substring(3,4)%];
            var<CP>{int} [%=i.name%]I(manager,[%=i.name%]_value_integrity);
            range [%=i.name%]_value_confidentiality = [%=i.valueConfidentiality.substring(1,2)]..
                          [%=i.valueConfidentiality.substring(3,4)%];
            var<CP>{int} [%=i.name%]C(manager,[%=i.name%]_value_confidentiality);
            range [%=i.name%]_value_availability = [%=i.valueAvailability.substring(1,2)]..
                          [%=i.valueAvailability.substring(3,4)%];
            var<CP>{int} [%=i.name%]A(manager,[%=i.name%]_value_availability);
            var<CP>(int) [%=i.name%](manager,1..100000);
            var<CP>(int) Risk[%=i.name%](manager,1..100000);
            [% var aux:String="";
            listaVariables.add(i.name);
            aux = i.name+"I";
            listaVariables.add(aux);aux = i.name+"C";
            listaVariables.add(aux);aux = i.name+"A";
            listaVariables.add(aux); aux = "Risk"+i.name;
            listaVariables.add(aux); %]
APPENDIX A. TRANSFORMATION TEMPLATE

listaVariablesAsset.add(aux);
listaVariablesAssetAlternative.add(i.name);
}
}

// Global variable for risk total of business process
var<CP>{int} RiskTotal_{%=p.name%}(manager,1..100000);
[%totalvariables = totalvariables+1;%]
[% var aux:String=""; aux = "RiskTotal_"+p.name;
listaVariables.add(aux);
listaVariablesAsset.add(aux);%]

// Risk-Vulnerabilities variables
[%for (a in AssessmentFlow.allInstances) { %]
[%if(a.source = p){%]
[ %for (r in a.target.risks) { %]
[ %totalvariables = totalvariables+2;%]
// Ranges
range {%=r.name.toLowerCase()%}_freq =
{%=r.frequency.substring(1,2)%}..
{%=r.frequency.substring(3,4)%};
range {%=r.name.toLowerCase()%}_conseq =
{%=r.consequence.substring(1,2)%}..
{%=r.consequence.substring(3,4)%};
// Variables
var<CP>{int} {%=r.name.toLowerCase()%}_frequency(manager,%{=r.name.toLowerCase()%}_freq);
var<CP>{int} {%=r.name.toLowerCase()%}_consequence(manager,%{=r.name.toLowerCase()%}_conseq);
[ % var aux:String=""; aux = r.name.toLowerCase()+'_frequency';
listaVariables.add(aux);
var aux:String=""; aux = r.name.toLowerCase()+'_consequence';
listaVariables.add(aux);
listaVariablesRisk.add(r.name);
%
};%
// Treatments variables
[%for (t in a.target.treatment) { %]
[ %totalvariables = totalvariables+1;%]
// Ranges
[ %if(t.riskReduction.length>3){%]
range {%=t.name.toLowerCase()%}_rr =
{%=t.riskReduction.substring(1,3)%}..
{%=t.riskReduction.substring(4,6)%};
[ %} else {%]
range {%=t.name.toLowerCase()%}_rr =
{%=t.riskReduction.substring(1,2)%}..
{%=t.riskReduction.substring(3,4)%};
[ %}
// Variables
var<CP>{int} {%=t.name.toLowerCase()%}_riskreduction(manager,%{=t.name.toLowerCase()%}_rr);
[ % var aux:String="";
aux = t.name.toLowerCase()+'_riskreduction';
listaVariables.add(aux);listaVariablesTreatment.add(aux);
%
}
%
]

// allVariables[1..{%=totalvariables%}];
[%for (i in Sequence(1..%{=totalvariables%}) ) { %]
[ allVariables[{%=i%}] = %{=listaVariables.at(i-1)%};
%
}%]

Integer np(manager.getNPropag());
cout << "Making Risk Assessment and Check Conformance" << endl;
Integer c(0);
minimize<manager>
RiskTotal_{%=p.name%}
subject to{
  var aux:String();
  for (la in listaVariablesAsset) {
    if("RiskTotal".isSubstringOf(la)) {
      manager.post("%=la%" == ("%=poolFormula.get(p)%") onBounds);
      manager.post("%=la%" > Acceptable_Risk*%listaVariablesAsset.size()-1, onBounds);
    } else {
      manager.post("%=listaVariablesAssetAlternative.at(listaVariablesAsset.indexOf(la))%" ==
        ("%=listaVariablesAssetAlternative.at(listaVariablesAsset.indexOf(la))%")I +
        ("%=listaVariablesAssetAlternative.at(listaVariablesAsset.indexOf(la))%")C +
        ("%=listaVariablesAssetAlternative.at(listaVariablesAsset.indexOf(la))%")A),
        onBounds);
    }
  }
}
using{
  labelFF(allVariables);
  cout << "***************************************" << " endl;
  for (i in Sequence{1..totalvariables}) {
    print("%=i%"._name = "%=listaVariablesAssetAlternative.at(listaVariablesAsset.indexOf(la))%")
  }
}
cout << "\#[\%=listaVariables.at(i-1)\%] = \" << [\%=listaVariables.at(i-1)\%] << endl;
System.output("[\%=listaVariables.at(i-1)\%]", [\%=listaVariables.at(i-1)\%]);
}

 cout << "Nb : " << c << endl;
 int tcpu = System.getCPUTime();
cout << "time: " << tcpu - t0 << endl;
cout << "\#choices = " << manager.getNChoice() << endl;
cout << "\#fail = " << manager.getNFail() << endl;
cout << "\#propag = " << manager.getNPropag() - np << endl;
cout << "RiskTotal_['%=p.name%'] = " << RiskTotal_['%=p.name%'].getValue() << endl;
%
%
// Auxiliar functions

// Determine number of Paths
operation Activity numPath() : Integer {
  var n : Integer = 0;
  for(s in SequenceFlow.allInstances){
    if(s.source.name.equals(self.name)){
      n = n+1;
    }
  }
  return n;
}

// Determine a set of paths
operation Activity computePath(graph : Map) : Sequence {
  var camino: Sequence();
  var aux := graph.get(self);
  camino.add(self);
  for(v in aux){
    aux.remove(v);
    camino.add(v);
    if(Gateway.allInstances.includes(v) and v.isOpen()){
      camino = camino;
    } else {
      for(w in graph.get(v)){
        aux.add(w);
        camino.add(w);
      }
    }
  }
  return camino;
}

// Determine auxiliar graphs
operation Pool generatedGraph() : Map {
  var graph: Map();
  var graphnames: Map();
  var graphi: Map();
  var start;
  var end;
  // Creating nodes
  for(v in self.vertex){
    graph.put(v, new Sequence);
    graphnames.put(v.name, new Sequence);
    graphi.put(v, new Sequence);
    if(Start.allInstances.includes(v)){
      start = v;
    }
  }

  // Auxiliar functions

  // Determine number of Paths
  operation Activity numPath() : Integer {
    var n : Integer = 0;
    for(s in SequenceFlow.allInstances){
      if(s.source.name.equals(self.name)){
        n = n+1;
      }
    }
    return n;
  }

  // Determine a set of paths
  operation Activity computePath(graph : Map) : Sequence {
    var camino: Sequence();
    var aux := graph.get(self);
    camino.add(self);
    for(v in aux){
      aux.remove(v);
      camino.add(v);
      if(Gateway.allInstances.includes(v) and v.isOpen()){
        camino = camino;
      } else {
        for(w in graph.get(v)){
          aux.add(w);
          camino.add(w);
        }
      }
    }
    return camino;
  }

  // Determine auxiliar graphs
  operation Pool generatedGraph() : Map {
    var graph: Map();
    var graphnames: Map();
    var graphi: Map();
    var start;
    var end;
    // Creating nodes
    for(v in self.vertex){
      graph.put(v, new Sequence);
      graphnames.put(v.name, new Sequence);
      graphi.put(v, new Sequence);
      if(Start.allInstances.includes(v)){
        start = v;
      }
    }
  }

if(End.allInstances.includes(v)){
  end = v;
}
}

// Creating edges
for(s in self.sequenceflow){
  if(self.vertex.indexOf(s.source)<>-1 and
      self.vertex.indexOf(s.target)<>-1) {
    if(not graph.get(s.source).includes(s.target)) {
      var h := graph.get(s.source);
      var h1 := graphnames.get(s.source.name);
      h.add(s.target);
      h1.add(s.target.name);
      graph.put(s.source, h);
      graphnames.put(s.source.name, h1);
    }
    // Bidirectional graph
    var h := graphi.get(s.target);
    h.add(s.source);
    graphi.put(s.target, h);
  }
}
return graph;
}

// Determine risk formula from an activity
operation Pool getFormula(start : Activity, graph : Map) : String{
  var last := start;
  var Q := graph.get(start);
  var aux := "";
  var visitados : Sequence();
  var niveles : Integer = 0;
  var numeroCaminos : Sequence(Integer);
  var nodoCamino : Map();
  var lastGateway;
  var camino : Sequence();
  var numPaths = 0;
  while(not Q.isEmpty()){ 
    var i := Q.first();
    Q.remove(i);
    visitados.add(i.name);
    if(i.isKindOf(Gateway)){
      if(not i.isOpen()){ // Open gateway
        niveles = niveles + 1;
        numPaths = i.numPath();
        numeroCaminos.add(numPaths);
        var num = 0;
        while(num < numPaths){
          camino = (graph.get(i).at(num)).computePath(graph);
          nodoCamino.put(num+1, camino);
          num = num + 1;
        }
      } else {
        if(i.isTypeOf(ParallelG)){
          aux := aux+"+";
        } else {
          // Do something
        }
      }
    } else {
      // Do something
    }
  }
  // Do something
  return aux;
}
APPENDIX A. TRANSFORMATION TEMPLATE

```java
aux := aux + max(";
}

if(i.isOpen()) { // Closing gateways
    numPaths = numPaths - 1;
    if(numeroCaminos.last() == niveles)
        aux := aux + ");
        niveles = 0;
    } else {
        niveles = niveles + 1;
    }
}

} else {
    if(Gateway.allInstances.includes(last) and not last.isOpen())
        aux = aux + "Risk" + i.name;
    else {
        if(last.isOpen() and numPaths > 0 and not last.isTypeOf(ParallelG))
            aux = aux + "," + "Risk" + i.name;
        else {
            if(Start.allInstances.includes(last))
                aux = aux + "Risk" + i.name;
            else {
                if(graph.get(i).size() > 0)
                    aux = aux + "+" + "Risk" + i.name;
            }
        }
    }
}

if(i.isKindOf(Gateway)) {
    if(numPaths > 0 ){
        Q = nodoCamino.get(numPaths);
    } else {
        for(x in graph.get(i)){
            if(not visitados.includes(x.name))
                Q.add(x);
        }
    }
    for(x in graph.get(i)){
        if(not visitados.includes(x.name))
            Q.add(x);
    }
    last = i;
}

return aux;
```
Appendix B

Example of CSP

Code gives in below corresponds to a piece of the COMET constraint model obtained for the transformation of *Human Resource* pool for the example in Figure 4.18. The other constraint models in the example are available to download in [Varela-Vaca 2012](#).

**COMET Constraint Model**

```csharp
// POOLNAME: Proceso1
import cotfd;

Solver<CP> manager();

int t0 = System.getCPUTime();

// Acceptable risk level
int Acceptable_Risk = 150;

// Taks variables
range A11_value_integrity = 1..2;
var<CP>{int} A11I(manager,A11_value_integrity);
range A11_value_confidentiality = 1..2;
var<CP>{int} A11C(manager,A11_value_confidentiality);
range A11_value_availability = 1..2;
var<CP>{int} A11A(manager,A11_value_availability);
var<CP>{int} A11(manager,1..100000);
var<CP>{int} RiskA11(manager,1..100000);

range A12_value_integrity = 1..3;
var<CP>{int} A12I(manager,A12_value_integrity);
range A12_value_confidentiality = 4..5;
var<CP>{int} A12C(manager,A12_value_confidentiality);
range A12_value_availability = 4..5;
var<CP>{int} A12A(manager,A12_value_availability);
var<CP>{int} A12(manager,1..100000);
var<CP>{int} RiskA12(manager,1..100000);

range A13_value_integrity = 1..1;
var<CP>{int} A13I(manager,A13_value_integrity);
range A13_value_confidentiality = 1..1;
var<CP>{int} A13C(manager,A13_value_confidentiality);
range A13_value_availability = 1..1;
var<CP>{int} A13A(manager,A13_value_availability);
```
APPENDIX B. EXAMPLE OF CSP

var<CP>{int} A13(manager,1..100000);
var<CP>{int} RiskA13(manager,1..100000);
range A14_value_integrity = 1..4;
var<CP>{int} A14I(manager,A14_value_integrity);
range A14_value_confidentiality = 1..1;
var<CP>{int} A14C(manager,A14_value_confidentiality);
range A14_value_availability = 1..1;
var<CP>{int} A14A(manager,A14_value_availability);

var<CP>{int} A14(manager,1..100000);
var<CP>{int} RiskA14(manager,1..100000);
range A15_value_integrity = 1..3;
var<CP>{int} A15I(manager,A15_value_integrity);
range A15_value_confidentiality = 4..5;
var<CP>{int} A15C(manager,A15_value_confidentiality);
range A15_value_availability = 1..1;
var<CP>{int} A15A(manager,A15_value_availability);

var<CP>{int} A15(manager,1..100000);
var<CP>{int} RiskA15(manager,1..100000);
range A16_value_integrity = 1..1;
var<CP>{int} A16I(manager,A16_value_integrity);
range A16_value_confidentiality = 1..1;
var<CP>{int} A16C(manager,A16_value_confidentiality);
range A16_value_availability = 1..1;
var<CP>{int} A16A(manager,A16_value_availability);

var<CP>{int} A16(manager,1..100000);
var<CP>{int} RiskA16(manager,1..100000);
range A17_value_integrity = 1..4;
var<CP>{int} A17I(manager,A17_value_integrity);
range A17_value_confidentiality = 1..4;
var<CP>{int} A17C(manager,A17_value_confidentiality);
range A17_value_availability = 1..4;
var<CP>{int} A17A(manager,A17_value_availability);

var<CP>{int} A17(manager,1..100000);
var<CP>{int} RiskA17(manager,1..100000);
range A18_value_integrity = 1..4;
var<CP>{int} A18I(manager,A18_value_integrity);
range A18_value_confidentiality = 1..4;
var<CP>{int} A18C(manager,A18_value_confidentiality);
range A18_value_availability = 1..4;
var<CP>{int} A18A(manager,A18_value_availability);

var<CP>{int} A18(manager,1..100000);
var<CP>{int} RiskA18(manager,1..100000);

// Global variable for risk total of business process
var<CP>{int} RiskTotal_Proceso1(manager,1..100000);

// Risk-Vulnerabilities variables
// Ranges
range r11_freq = 1..3;
range r11_conseq = 2..5;
// Variables
var<CP>{int} r11_frequency(manager,r11_freq);
var<CP>{int} r11_consequence(manager,r11_conseq);
// Ranges
range r12_freq = 3..4;
range r12_conseq = 4..5;
// Variables
var<CP>{int} r12_frequency(manager,r12_freq);
var<CP>{int} r12_consequence(manager,r12_conseq);
// Ranges
range r13_freq = 1..4;
range r13_conseq = 1..4;
// Variables
var<CP>{int} r13_frequency(manager,r13_freq);
var<CP>{int} r13_consequence(manager,r13_conseq);
// Treatments variables
//Ranges
range t11_rr = 10..30;
//Variables
var<CP>{int} t11_riskreduction(manager,t11_rr);
//Ranges
range t12_rr = 10..20;
//Variables
var<CP>{int} MaxCSP_Proceso1(manager,1..3);
var<CP>{int} RiskF1_Proceso1(manager,1..10000);
var<CP>{int} RiskF2_Proceso1(manager,1..10000);
var<CP>{int} RiskF3_Proceso1(manager,1..10000);
var<CP>{bool} f1_Proceso1(manager);
var<CP>{bool} f2_Proceso1(manager);
var<CP>{bool} f3_Proceso1(manager);
var<CP>{int} allVariables[1..56];
allVariables[1] = A11;
allVariables[2] = A11I;
allVariables[4] = A11A;
allVariables[5] = RiskA11;
allVariables[6] = A12;
allVariables[7] = A12I;
allVariables[8] = A12C;
allVariables[9] = A12A;
allVariables[10] = RiskA12;
allVariables[12] = A13I;
allVariables[14] = A13A;
allVariables[15] = RiskA13;
allVariables[16] = A14;
allVariables[17] = A14I;
allVariables[18] = A14C;
allVariables[19] = A14A;
allVariables[20] = RiskA14;
allVariables[21] = A15;
allVariables[22] = A15I;
allVariables[23] = A15C;
allVariables[24] = A15A;
allVariables[25] = RiskA15;
allVariables[26] = A16;
allVariables[27] = A16I;
allVariables[28] = A16C;
allVariables[29] = A16A;
allVariables[30] = RiskA16;
allVariables[31] = RiskTotal_Proceso1;
allVariables[32] = r11_frequency;
allVariables[33] = r11_consequence;
allVariables[34] = r12_frequency;
allVariables[35] = r12_consequence;
allVariables[36] = r13_frequency;
allVariables[37] = r13_consequence;
allVariables[38] = t11_riskreduction;
allVariables[39] = t12_riskreduction;
allVariables[40] = MaxCSP_Proceso1;
allVariables[41] = RiskF1_Proceso1;
allVariables[42] = RiskF2_Proceso1;
allVariables[43] = f1_Proceso1;
allVariables[44] = f2_Proceso1;
allVariables[45] = A17;
allVariables[46] = A17I;
allVariables[47] = A17C;
allVariables[48] = A17A;
allVariables[49] = RiskA17;
allVariables[50] = A18;
allVariables[51] = A18I;
allVariables[52] = A18C;
allVariables[53] = A18A;
allVariables[54] = RiskA18;
allVariables[55] = f3_Proceso1;
allVariables[56] = RiskF3_Proceso1;

Integer np(manager.getNPropag());

cout << "Making Risk Assessment and Check Conformance" << endl;

Integer c(0);
maximize<manager>
MaxCSP_Proceso1
subject to{

    manager.post(A11 == (A11I + A11C + A11A), onBounds);
    manager.post(RiskA11 == (A11* (r11_frequency
    -( r11_frequency * t11_riskreduction/100 ))*(r11_consequence
    -( r11_consequence * t11_riskreduction/100 ))), onBounds);

    manager.post(A12 == (A12I + A12C + A12A), onBounds);
    manager.post(RiskA12 == (A12* (r12_frequency
    -( r12_frequency * t11_riskreduction/100 ))*(r12_consequence
    -( r12_consequence * t11_riskreduction/100 )))*(A12*
    (r12_frequency
    -( r12_frequency * t12_riskreduction/100 )))*(r12_consequence
    -( r12_consequence * t12_riskreduction/100 )) + (A12*
    (r13_frequency
    -( r13_frequency * t12_riskreduction/100 )))*(r13_consequence
    -( r13_consequence * t12_riskreduction/100 )))
    ), onBounds);
manager.post(A13 == (A13I + A13C + A13A),onBounds);
manager.post(RiskA13 == (A13\times(r_{11,\text{frequency}}-
(r_{11,\text{frequency}}\times t_{11,\text{riskreduction}}/100))\times(r_{11,\text{consequence}}-
(r_{11,\text{consequence}}\times t_{11,\text{riskreduction}}/100))),onBounds);

manager.post(A14 == (A14I + A14C + A14A),onBounds);
manager.post(RiskA14 == (A14\times(r_{11,\text{frequency}}-
(r_{11,\text{frequency}}\times t_{11,\text{riskreduction}}/100))\times(r_{11,\text{consequence}}-
(r_{11,\text{consequence}}\times t_{11,\text{riskreduction}}/100)))+(A14\times(r_{12,\text{frequency}}-
(r_{12,\text{frequency}}\times t_{11,\text{riskreduction}}/100))\times(r_{12,\text{consequence}}-
(r_{12,\text{consequence}}\times t_{11,\text{riskreduction}}/100)))+(A14\times(r_{13,\text{frequency}}-
(r_{13,\text{frequency}}\times t_{12,\text{riskreduction}}/100))\times(r_{13,\text{consequence}}-
(r_{13,\text{consequence}}\times t_{12,\text{riskreduction}}/100))),onBounds);

manager.post(A15 == (A15I + A15C + A15A),onBounds);
manager.post(RiskA15 == (A15\times(r_{11,\text{frequency}}-
(r_{11,\text{frequency}}\times t_{11,\text{riskreduction}}/100))\times(r_{11,\text{consequence}}-
(r_{11,\text{consequence}}\times t_{11,\text{riskreduction}}/100)))+(A15\times(r_{12,\text{frequency}}-
(r_{12,\text{frequency}}\times t_{11,\text{riskreduction}}/100))\times(r_{12,\text{consequence}}-
(r_{12,\text{consequence}}\times t_{11,\text{riskreduction}}/100)))+(A15\times(r_{13,\text{frequency}}-
(r_{13,\text{frequency}}\times t_{12,\text{riskreduction}}/100))\times(r_{13,\text{consequence}}-
(r_{13,\text{consequence}}\times t_{12,\text{riskreduction}}/100))},onBounds);
APPENDIX B. EXAMPLE OF CSP

```plaintext
r13_consequence * t12_riskreduction/100
})
), onBounds);
manager.post(A16 == (A16I + A16C + A16A), onBounds);
manager.post(RiskA16 == (A16*
(r11_frequency
-(
 r11_frequency * t11_riskreduction/100
 )*(r11_consequence
-(
 r11_consequence * t11_riskreduction/100
 )))*(A16*
(r12_frequency
-(
 r12_frequency * t11_riskreduction/100
 ))*(r12_consequence
-(
 r12_consequence * t11_riskreduction/100
 )))+(A16*
(r13_frequency
-(
 r13_frequency * t12_riskreduction/100
 ))*(r13_consequence
-(
 r13_consequence * t12_riskreduction/100
 ))), onBounds);
manager.post(A17 == (A17I + A17C + A17A), onBounds);
manager.post(RiskA17 == (A17*
(r11_frequency
-(
 r11_frequency * t11_riskreduction/100
 ))*(r11_consequence
-(
 r11_consequence * t11_riskreduction/100
 )))*(A17*
(r13_frequency
-(
 r13_frequency * t12_riskreduction/100
 ))*(r13_consequence
-(
 r13_consequence * t12_riskreduction/100
 ))), onBounds);
manager.post(A18 == (A18I + A18C + A18A), onBounds);
manager.post(RiskA18 == (A18*
(r13_frequency
-(
 r13_frequency * t12_riskreduction/100
 ))*(r13_consequence
-(
 r13_consequence * t12_riskreduction/100
 ))), onBounds);
manager.post(RiskF1_Proceso1 == (RiskA11+RiskA12+RiskA17+RiskA18)/4, onBounds);
manager.post(RiskF2_Proceso1 == (RiskA11+RiskA14+RiskA18)/3, onBounds);
manager.post(RiskF3_Proceso1 == (RiskA11+RiskA13+RiskA15+RiskA16+RiskA18)/3, onBounds);
manager.post(f1_Proceso1 == (RiskF1_Proceso1 > Acceptable_Risk), onBounds);
manager.post(f2_Proceso1 == (RiskF2_Proceso1 > Acceptable_Risk), onBounds);
manager.post(f3_Proceso1 == (RiskF3_Proceso1 > Acceptable_Risk), onBounds);
```
manager.post(MaxCSP_Proceso1 == (f1_Proceso1+f2_Proceso1+f3_Proceso1));

using {
    labelFF(allVariables);
    cout << "*******************************************************************************" << endl;
    cout << "#A11 = " << A11 << endl;
    System.output("A11", A11);
    cout << "#A11I = " << A11I << endl;
    System.output("A11I", A11I);
    cout << "#A11C = " << A11C << endl;
    System.output("A11C", A11C);
    cout << "#A11A = " << A11A << endl;
    System.output("A11A", A11A);
    cout << "#RiskA11 = " << RiskA11 << endl;
    System.output("RiskA11", RiskA11);
    cout << "#A12 = " << A12 << endl;
    System.output("A12", A12);
    cout << "#A12I = " << A12I << endl;
    System.output("A12I", A12I);
    cout << "#A12C = " << A12C << endl;
    System.output("A12C", A12C);
    cout << "#A12A = " << A12A << endl;
    System.output("A12A", A12A);
    cout << "#RiskA12 = " << RiskA12 << endl;
    System.output("RiskA12", RiskA12);
    cout << "#A13 = " << A13 << endl;
    System.output("A13", A13);
    cout << "#A13I = " << A13I << endl;
    System.output("A13I", A13I);
    cout << "#A13C = " << A13C << endl;
    System.output("A13C", A13C);
    cout << "#A13A = " << A13A << endl;
    System.output("A13A", A13A);
    cout << "#RiskA13 = " << RiskA13 << endl;
    System.output("RiskA13", RiskA13);
    cout << "#A14 = " << A14 << endl;
    System.output("A14", A14);
    cout << "#A14I = " << A14I << endl;
    System.output("A14I", A14I);
    cout << "#A14C = " << A14C << endl;
    System.output("A14C", A14C);
    cout << "#A14A = " << A14A << endl;
    System.output("A14A", A14A);
    cout << "#RiskA14 = " << RiskA14 << endl;
    System.output("RiskA14", RiskA14);
    cout << "#A15 = " << A15 << endl;
    System.output("A15", A15);
    cout << "#A15I = " << A15I << endl;
    System.output("A15I", A15I);
    cout << "#A15C = " << A15C << endl;
    System.output("A15C", A15C);
    cout << "#A15A = " << A15A << endl;
    System.output("A15A", A15A);
    cout << "#RiskA15 = " << RiskA15 << endl;
    System.output("RiskA15", RiskA15);
    cout << "#A16 = " << A16 << endl;
    System.output("A16", A16);
    cout << "#A16I = " << A16I << endl;
    System.output("A16I", A16I);
    cout << "#A16C = " << A16C << endl;
    System.output("A16C", A16C);
    cout << "#A16A = " << A16A << endl;
System.output("A16", A16);
cout << "#RiskA16 = " << RiskA16 << endl;
System.output("A17", A17);
cout << "#A17I = " << A17I << endl;
System.output("A17I", A17I);
cout << "#A17C = " << A17C << endl;
System.output("A17C", A17C);
cout << "#A17A = " << A17A << endl;
System.output("A17A", A17A);
cout << "#RiskA17 = " << RiskA17 << endl;
System.output("A18", A18);
cout << "#A18I = " << A18I << endl;
System.output("A18I", A18I);
cout << "#A18C = " << A18C << endl;
System.output("A18C", A18C);
cout << "#A18A = " << A18A << endl;
System.output("A18A", A18A);
cout << "#RiskA18 = " << RiskA18 << endl;
}
cout << "Nb : " << c << endl;
int tcpu = System.getCPUTime();
cout << "time: " << tcpu - t0 << endl;
cout << "#choices = " << manager.getNChoice() << endl;
cout << "#fail = " << manager.getNFail() << endl;
cout << "#propag = " << manager.getNPropag() - np << endl;
Appendix C

Validation constraints for business process models

In Chapter 4, OPBUS tool has been presented (Varela Vaca, 2012). This tool is equipped with a BPMN 2.0 modeller extended to support the risk extension proposed. The editor has been provided with validation capabilities (Varela-Vaca et al., 2010b). A validation consists in a set of constraints developed using Epsilon Validation Language (EVL) (Kolovos, 2012). Those constraints are evaluated during design of a business process models. This validations enable ensure the business process desing by identifying structural faults (Huang et al., 2008) such as live locks, starvations, lack of synchronization, etc. Validations are composed of two parts: (1) check section which return true if the validation is correct and false otherwise; (2) message section which indicates the cause of error. A set of the most representative validation constraints implemented in OPBUS tool are given below.

LiveLock

The main idea, subjacent in this algorithm is the detection of cycles in a model. The algorithm makes an in depth of traverse of the business process diagram, as if there is re-visited node detection. A path in reverse order is constructed from the parent of this node. If this path includes the node in question, then there is a cycle, otherwise there is no cycle.

```plaintext
constraint liveLock{
    check{
        var path := new OrderedSet;
        var auxiliar := new List;
        var visited := new List;
        var neighbors := new List;
        var last;
        var neighbor;
        for(s in Start){
            if(self.vertexs.includes(s)){
                path.add(s);
                var i := 0;
                while(i<path.size()){
                    neighbor := path.at(i);
                    neighbors := graph.get(neighbor);
                    for(neighbor in neighbors){
                        if(self.vertexs.includes(neighbor)){
                            auxiliar.add(neighbor);
                        }
                    }
                    visited.add(s);
                    if(path.includes(neighbor)){
                        return false;
                    }
                    last := neighbor;
                }
            }
        }
        return true;
    }
}
```
visited.add(neighbor);
for(v in neighbors){
    if(visited.includes(v)){
        var pathAuxiliar := new List;
        var index := path.indexOf(neighbor)-1;
        while(not (auxiliar.at(index)=="*")){
            pathAuxiliar.add(auxiliar.at(index));
            index := path.indexOf(auxiliar.at(index))-1;
        }
        if(pathAuxiliar.includes(v)){
            last := v;
            return false;
        }
    } else {
        if(i==0){
            auxiliar.add("*");
        } else{
            auxiliar.add(neighbor);
        }
        path.add(v);
    }
}
i:=i+1;
return true;
message: "Livelock:" + last.name + ", " + neighbor.name + ")"}

Starvation
In this case the same idea of traversing of the diagram is implicit in the algorithms of Starvation and Lack of synchronization. The most significant variation between these algorithms is focused on the detection of particular gateways (inclusive or exclusive). Each gateway acting as splitter of the paths has been associated with a "scope". This scope stores the gateway (merger gateway) that closed the paths opened before. Therefore, when if we find two pairs of gateways whereby the first one is a splitter and the second one is a merger, then if the first is an exclusive gateway and the second is an inclusive gateway then Starvation has been located. In the case of an inclusive gateway as the splitter and an exclusive gateway as the merger, then Lack of synchronization is found.

constraint starvation{
    check{
        var m := new Native('bpmnfmea.tools.validation');
        var path := new OrderedSet;
        var auxiliar := new List;
        var neighbors := new List;
        var envs := new Map;
        var arr;
        var arr2;
        var arr3;
        var neighbor;
        for(s in Start){
            if(self.vertexs.includes(s)){
                path.add(s);
                var i := 0;
                while(i<path.size()){
neighbor := path.at(i);
neighbors := graph.get(neighbor);

if(m.isExclusiveDataBasedGatewayType(neighbor) or
m.isExclusiveEventBasedGatewayType(neighbor)){
    envs.put(neighbor, new OrderedSet);
}

for(v in neighbors){
    if(m.isInclusiveDataBasedGatewayType(v) or
m.isInclusiveEventBasedGatewayType(v)){
        var pathAuxiliar := new List;
        var index := path.indexOf(neighbor)-1;
        while(not (auxiliar.at(index)=="*")){
            if(m.isExclusiveDataBasedGatewayType(auxiliar.at(index)) or
m.isExclusiveEventBasedGatewayType(auxiliar.at(index))){
                arr := v;
                arr2 := auxiliar.at(index);
                return false;
            }
            pathAuxiliar.add(auxiliar.at(index));
            index := path.indexOf(auxiliar.at(index))-1;
        }
    } else {
        if(i==0){
            auxiliar.add("*");
        } else{
            auxiliar.add(neighbor);
        }
        path.add(v);
    }
    i:=i+1;
}

return true;
}
message: "Starvation: {" + arr.name + "," + arr2.name + "}"}

Lack of Synchronization for Gateways

This algorithm is based on the same ideas as those of the starvation algorithm: the traversing of the diagram is similar but it uses the concept of scope. However another intrinsic problem appears in form of the nesting of parallels. In the case of several parallels being opened and then not closed correctly, the detection of the subjacent fault it causes is not a trivial problem.

constraint lacksynchronous{
    check{
        var m := new Native('bpmnfmea.tools.validation');
        var path := new OrderedSet;
        var auxiliar := new List;
        var visitados := new List;
        var neighbors := new List;
        var envs := new Map;
        var arr;
APPENDIX C. VALIDATION CONSTRAINTS

```javascript
var arr2;
var arr3;
var neighbor;
for(s in Start){
    if(self.vertexs.includes(s)){
        path.add(s);
        var i := 0;
        while(i<path.size()){ // 
            neighbor := path.at(i);
            neighbors := graph.get(neighbor);
            visitados.add(neighbor);
            if(m.isInclusiveDataBasedGatewayType(neighbor) or
               m.isInclusiveDataBasedGatewayType(neighbor)){
                envs.put(neighbor, new OrderedSet);
            }
            for(v in neighbors){
                if(m.isExclusiveDataBasedGatewayType(neighbor) or
                   m.isExclusiveEventBasedGatewayType(neighbor)){
                    var pathAuxiliar := new List;
                    var index := path.indexOf(neighbor)-1;
                    while(not (auxiliar.at(index)=="*")){
                        if(m.isInclusiveDataBasedGatewayType(auxiliar.at(index)) or
                           m.isInclusiveDataBasedGatewayType(auxiliar.at(index))){
                            arr := v;
                            arr2 := auxiliar.at(index);
                            return false;
                        }
                        pathAuxiliar.add(auxiliar.at(index));
                        index := path.indexOf(auxiliar.at(index))-1;
                    }
                    pathAuxiliar.add(auxiliar.at(index));
                    index := path.indexOf(auxiliar.at(index))-1;
                }
                else {
                    if(i==0){
                        auxiliar.add("*");
                    } else{
                        auxiliar.add(neighbor);
                    }
                    path.add(v);
                }
                i:=i+1;
            }
        }
    }
}
return true;
}
message: "Lack synchronous: "+ arr.name + "," + arr2.name +")"
```

Start Events without incoming Sequence Flows

This algorithm checks if the star events (excluding start message events) do not receive any sequence flow as input.

case dontHaveTargetFlows{
    check{
        for (p in SequenceEdge)
```
Checking Correctness of Time Event Conditions

This algorithm checks the correctness of the condition in the time events.

```java
constraint startTimerEventDoesnotHaveCorrectTime{
    check{
        var m := new Native('bpmnfmea.tools.validation');
        if(m.isTimeStartType(self)){
            if(not self.time.isDefined()){
                return false;
            } else {
                return m.isCorrectTime(self.time);
            }
        } else {
            return true;
        }
    }
    message: 'Time property have to be changed for a correct value'
}
```

Incoming message flow in Star Message Event

This algorithm checks that any start message event receives at least one start event.

```java
constraint starMessageDoesnotHaveMessageEvent{
    check{
        var m := new Native('bpmnfmea.tools.validation');
        if(m.isMessageStartType(self)){
            for(msg in MessageEdge){
                if(msg.target == self){
                    return true;
                }
            }
            return false;
        } else {
            return true;
        }
    }
    message: 'Start message needs an incoming message'
}
```

Isolated Activity

This algorithm checks that there is any isolated activity, that is, the activity is not connected with other elements in the model.

```java
constraint noConnected{
    check{
        var ps = new OrderedSet;
```
```javascript
var pt = new OrderedSet;
for(r in SequenceEdge){
    if(r.source == self){
        ps.add(r);
    }
    if(r.target == self){
        pt.add(r);
    }
}
if(ps.size() == 0 and pt.size() == 0){
    return false;
} else {
    return true;
}
message: 'This activity is isolated'
}

No Sequence Flows Connecting Pools

This algorithm checks that there is no sequence flows between two different pools.

```javascript
constraint flowBetweenPools{
    check{
        return self.source.getPool() == self.target.getPool();
    }
    message: 'Flow edges join two different pools'
}
```
Appendix D

Feature Analysis of SSL and TLS protocol

In this appendix, we present the analysis of SSL and TLS protocols corresponding with SSL/TLS feature model shown in Section 5.4.2 of Chapter 5.

Transport Layer Security (TLS) protocol and Secure Socket Layer (SSL) (hereinafter SSL/TLS) enable the authentication of client and server by providing signatures (certificates or passwords). Furthermore, data might be signed and ciphered in order to achieve data confidentiality and data integrity.

SSL/TLS provides three main security services: confidentiality, by encrypting data; message integrity, by using a message authentication code (MAC); authentication, by using digital signatures. SSL/TLS allows the authentication of both parties, server authentication with an unauthenticated client, and total anonymity. The authentication of client and server might be carried out through digital signatures. Nowadays, these digital signatures are mostly based on certificates (i.e. X.509 standard) or shared keys. Certificates are based on the concept of public key cryptography known as asymmetric cryptography. This method generates two keys: a public key and private key. If one key is used to encrypt a message (signed and ciphered) then the other must be used to decrypt it. This makes it possible to receive secure messages by simply publishing one key (the public key) and keeping the other secret (the private key). Anyone may encrypt a message using the public key, but only the owner of the private key will be able to decrypt and read it. In the case of using certificates, these always have to be verified to ensure proper signing by a trusted Certificate Authority (CA). On the other hand, these protocols also provide anonymous authentication by using Diffie-Hellman for key exchange from SSLv3.0, TLSv1.0 and earlier versions.

SSL/TLS protocol is based on a handshake sequence (see Figure D.1) whose main features, as used by the client and server, are listed below:

1. Negotiate the Cipher Suite to be used during data transfer, and exchange random numbers (master key).
2. Establish and share a Session ID between client and server.
3. Optionally authenticate the server to the client.
4. Optionally authenticate the client to the server.

The first step, cipher suite negotiation, allows the client and server to choose a Cipher Suite supported by both of them. For instance, the SSLv3.0 protocol specification defines 31 Cipher Suites. A Cipher Suite is defined by the following components:

- **Key Exchange Method (KEM)** defines which cryptographic algorithms have been used to generate cryptographic keys. For instance, SSLv2.0 uses RSA key exchange only, while SSLv3.0 supports a choice of key exchange algorithms including the RSA key exchange when certificates are used, and Diffie-Hellman key exchange for exchanging keys without certificates and without prior communication between client and server.

- **Cipher for Data Transfer** defines which conventional cryptographic algorithms are used to encrypt the message in the transmission. For instance, SSLv2.0 supports a reduced number of ciphers such as RC2, RC4 and Data Encryption Standard (DES). However, SSLv3.0 and earlier versions of TLS increase the number of cryptographic algorithms supported.

- **Message Digest** is used to create a Message Authentication Code (MAC) which is
encrypted with the message to provide integrity and to prevent replay attacks. For instance, SSLv2.0 only supports MD5, while SSLv3.0 and earlier versions of TLS support other methods such as SHA-1.

D.1 Constraint Model for Feature Model of SSL/TLS

The code in below represent the formalization by constraint programming of the feature model shown in Figure 5.8 shown in Chapter 5. This model together previous ones are available to download in [Varela-Vaca, 2012].

```cpp
import cotfd;

/********* Configuration *************/
Solver<CP> manager(); // Solver manager
SolutionPool solutions(); // Pool of solutions
ofstream outfile("c:\outfile.txt");

int startTime = System.getWCTime(); // Set up begining time
int threshold = 3000;

AbstractSearchController searchController = manager.getSearchController();

// Solutions are checked if feasible
whenever searchController@onFeasibleSolution(Solution s){
    // cout << "#SSLtime: " << System.getWCTime() - startTime << " ms" << endl;
    // solutions is stored in the pool of solutions
    solutions.add(s);
}

/********* Types *************/
enum Protocols = {SSLv2, SSLv3, TLSv1, TLSv11, TLSv12};
enum MAC = {NoDigest, MD5, SHA1, SHA256, SHA512};
enum KeyExchangeMethod = {RSA, DH_DSS, DH_RSA, DHE_DSS, DHE_RSA, DH_anon, Fortezza};

enum Enc_Type = {NoEnc, RC440, RC4128, RC4128EXP, AES128, AES192, AES256, RC2128, RC2128EXP, RC240, DES, DES40, DES56, DES64, DES192, TripleDES168, IDEA128, Fortezza96};

/********* Ranges/Variables/Constraints CP *************/
range literal = 0..1;
range aro_reduction = 0..100;

//Non-functional requirements
var<CP>{int} confidentiality(manager, literal);
var<CP>{int} integrity(manager, literal);
var<CP>{int} authentication(manager, literal);
var<CP>{int} secure_protocol(manager, literal);
var<CP>{int} ssl_tls(manager, literal);
var<CP>{int} protocol(manager, Protocols);
var<CP>{int} ciphersuite(manager, literal);
var<CP>{int} sessionkey(manager, literal);
var<CP>{int} authenticationmethod(manager, literal);
```

...
<table>
<thead>
<tr>
<th>Protocol</th>
<th>KEM</th>
<th>Cipher</th>
<th>MAC</th>
<th>Master Key</th>
<th>Key-size limit</th>
<th>Session ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSLv2.0</td>
<td>RSA</td>
<td>RC4 128 bits</td>
<td>MD5</td>
<td>256 bits</td>
<td>Minimum 512 bits</td>
<td>16 bytes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RC4 128 bits Export</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RC2 128 bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RC2 128 bits Export</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>IDEA 128 bits</td>
<td>MD5</td>
<td>256 bits</td>
<td>Minimum 512 bits</td>
<td>16 bytes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DES 64 bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DES 192 bits EDE3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSLv3.0</td>
<td>Null</td>
<td>Null</td>
<td>Null</td>
<td>384 bits</td>
<td>Minimum 512 bits</td>
<td>32 bytes</td>
</tr>
<tr>
<td></td>
<td>RSA</td>
<td>RC4 128 bits</td>
<td>MD5</td>
<td>256 bits</td>
<td>Minimum 512 bits</td>
<td>16 bytes</td>
</tr>
<tr>
<td></td>
<td>DH with DSS</td>
<td>RC4 128 bits</td>
<td>SHA</td>
<td>256 bits</td>
<td>Minimum 512 bits</td>
<td>16 bytes</td>
</tr>
<tr>
<td></td>
<td>DH with RSA</td>
<td>RC4 128 bits Export</td>
<td></td>
<td>256 bits</td>
<td>Minimum 512 bits</td>
<td>16 bytes</td>
</tr>
<tr>
<td></td>
<td>DHE with DSS</td>
<td>RC2 128 bits</td>
<td>SHA</td>
<td>256 bits</td>
<td>Minimum 512 bits</td>
<td>16 bytes</td>
</tr>
<tr>
<td></td>
<td>DHE with RSA</td>
<td>RC2 128 bits</td>
<td>SHA</td>
<td>256 bits</td>
<td>Minimum 512 bits</td>
<td>16 bytes</td>
</tr>
<tr>
<td></td>
<td>DH with Anonymous</td>
<td>RC2 128 bits Export</td>
<td></td>
<td>256 bits</td>
<td>Minimum 512 bits</td>
<td>16 bytes</td>
</tr>
<tr>
<td></td>
<td>Fortezza</td>
<td>IDEA 128 bits</td>
<td>MD5</td>
<td>256 bits</td>
<td>Minimum 512 bits</td>
<td>16 bytes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DES 56 bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DES 64 bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DES 192 bits EDE3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DES 40 bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fortezza 96 bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3DES 168 bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table D.1 – Summarization of feature related to SSL

*Authentication by certificate only RSA and MD5.*
<table>
<thead>
<tr>
<th>Protocol</th>
<th>KEM</th>
<th>Cipher</th>
<th>MAC</th>
<th>Master Key</th>
<th>Key-size limit</th>
<th>Session ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLSv1.0</td>
<td>Null</td>
<td>Null</td>
<td>Null</td>
<td>384 bits</td>
<td>Minimum 512 bits</td>
<td>32 bytes</td>
</tr>
<tr>
<td>RSA</td>
<td>RC4 40 bits</td>
<td>MD5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DH with DSS</td>
<td>RC4 128 bits</td>
<td>SHA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DH with RSA</td>
<td>RC2 40 bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DHE with DSS</td>
<td>IDEA 128 bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DHE with RSA</td>
<td>DES 56 bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DH with Anonymous</td>
<td>DES 40 bits</td>
<td>3DES 168 bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>KRB5</td>
<td>AES 128 bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>AES 256 bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TLSv1.1</td>
<td>Null</td>
<td>Null</td>
<td>Null</td>
<td>384 bits</td>
<td>Minimum 512 bits</td>
<td>32 bytes</td>
</tr>
<tr>
<td>RSA</td>
<td>RC4 40 bits</td>
<td>MD5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DH with DSS</td>
<td>RC4 128 bits</td>
<td>SHA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DH with RSA</td>
<td>RC2 40 bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DHE with DSS</td>
<td>IDEA 128 bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DHE with RSA</td>
<td>DES 56 bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DH with Anonymous</td>
<td>DES 40 bits</td>
<td>3DES 168 bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KRB5</td>
<td></td>
<td>AES 128 bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>AES 256 bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TLSv1.2</td>
<td>Null</td>
<td>Null</td>
<td>Null</td>
<td>384 bits</td>
<td>None</td>
<td>32 bytes</td>
</tr>
<tr>
<td>RSA</td>
<td>RC4 128 bits</td>
<td>MD5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DH with DSS</td>
<td>3DES 168 bits</td>
<td>SHA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DH with RSA</td>
<td>AES 128 bits</td>
<td>SHA-224</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DHE with DSS</td>
<td>AES 256 bits</td>
<td>SHA-256</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DHE with RSA</td>
<td></td>
<td>SHA-384</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DH with Anonymous</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RSA PSK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ECDH with EDSA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ECDH with RSA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table D.2 – Summarization of feature related to TLS
var<CF>{int} keychangemethod(manager,KeyExchangeMethod);
var<CF>{int} cipher_enc_type(manager,Enc_Type);
var<CF>{int} mac(manager,NAC);

var<CF>{int} certificate(manager,literal);
var<CF>{int} x509(manager,literal);
var<CF>{int} OpenPGP(manager,literal);
var<CF>{int} digital_signature(manager,literal);
var<CF>{int} SRP(manager,literal);
var<CF>{int} PSH(manager,literal);
var<CF>{int} Anonymous(manager,literal);

// ALE, ARQ y SLE data calculation
var<CF>{int} ssl_tls_aroreduction(manager,aro_reduction);
var<CF>{int} ssl_tls_aroreduction_ARP(manager,aro_reduction);
var<CF>{int} ssl_tls_aroreduction_certificate(manager,aro_reduction);
var<CF>{int} ssl_tls_aroreduction_certificate_x509(manager,aro_reduction);
var<CF>{int} ssl_tls_aroreduction_certificate_OpenPGP(manager,aro_reduction);
var<CF>{int} ssl_tls_sle(manager,0..10000);
var<CF>{int} ssl_tls_sle(manager,10..10);
var<CF>{int} ssl_tls_sle(manager,1000..1000);

range cost_range = 0..100;

// Cost data calculation
var<CF>{int} ssl_tls_protocol_cost(manager,cost_range);
var<CF>{int} ssl_digitalsignature_cost(manager,cost_range);
var<CF>{int} ssl_tls_certificate_cost(manager,cost_range);
var<CF>{int} ssl_tls_certificate_x509_cost(manager,cost_range);
var<CF>{int} ssl_tls_certificate_OpenPGP_cost(manager,cost_range);
var<CF>{int} ssl_tls_mac_cost(manager,cost_range);
var<CF>{int} ssl_tls_cipherenc_cost(manager,cost_range);
var<CF>{int} ssl_tls_keychangemethod_cost(manager,cost_range);
var<CF>{int} ssl_tls_cost(manager,0..500);

/*********** Solver *************/
Integer np(manager.getNPropag());
cout << "Solving feature model: " << endl;

// manager.limitTime(20);
// maximize<manager>
// minimize<manager>
// ssl_tls_ale
// ssl_tls_aroreduction
// ssl_tls_cost

solveall<manager> { // En caso de calculo de todas las soluciones posibles

// subject to // Comentar en caso de calcular de todas las soluciones posibles
manager.post(((digital_signature == 1 || certificate == 1)
&& keychangemethod > 0) ~> (authentication == 1 && confidentiality == 1));
manager.post(mac > 0 ~> (integrity == 1));

// Feature model
//manager.post(secure_protocol == 1 ~> confidentiality == 1
//&& integrity == 1);
manager.post(ssl_tls == 1 ~> secure_protocol == 1);
manager.post(secure_protocol == 1 ~> ssl_tls == 1);
manager.post(sessionkey == 1 ~> ssl_tls == 1);
manager.post(ssl_tls == 1 ~> sessionkey == 1);
D.1. CONSTRAINT MODEL FOR FEATURE MODEL OF SSL/TLS

manager.post((protocol == SSLv2 || protocol == SSLv3 || protocol == TLSv1 ||
  protocol == TLSv11 || protocol == TLSv12) ~> ssl tls == 1);
manager.post(ssl_tls == 1 ~> (protocol == SSLv2 || protocol == SSLv3
  || protocol == TLSv1 || protocol == TLSv11 || protocol == TLSv12));
manager.post(ciphersuite == 1 ~> ssl_tls == 1);
manager.post(authenticationmethod == 1 ~> ssl_tls == 1);
manager.post(ssl_tls == 1 ~> ciphersuite == 1);
manager.post(ssl_tls == 1 ~> authenticationmethod == 1);
manager.post(authenticationmethod == 1 ~> ((digital_signature == 1
  && certificate == 0) || (digital_signature == 0 && certificate == 1)));
manager.post(digital_signature == 1 ~> authenticationmethod == 1);
manager.post(certificate == 1 ~> (x509 == 1 || OpenPGP == 1)
  && !(x509 == 1 && OpenPGP == 1));
manager.post((x509 == 1 || OpenPGP == 1) ~> certificate == 1
  && !(x509 == 1 && OpenPGP == 1));
manager.post(certificate == 1 ~> (x509 == 1 || OpenPGP == 1)
  && !(x509 == 1 && OpenPGP == 1));
manager.post((keychangemethod == RSA || keychangemethod == DH_RSA
  || keychangemethod == DH_DSS || keychangemethod == DH_anon
  || keychangemethod == Fortezza) ~> ciphersuite == 1);
manager.post((keychangemethod == RSA || keychangemethod == DH_RSA
  || keychangemethod == DH_DSS || keychangemethod == DH_anon
  || keychangemethod == Fortezza)));
manager.post((mac == NoDigest || mac == MD5
  || mac == SHA1 || mac == SHA256 || mac == S12));
manager.post((cipher_enctype == NoEnc || cipher_enctype == RC4128
  || cipher_enctype == AES128 || cipher_enctype == AES192
  || cipher_enctype == AES256 || cipher_enctype == RC4128
  || cipher_enctype == DE540 || cipher_enctype == DE566
  || cipher_enctype == TripleDES168 || cipher_enctype == IDEA128
  || cipher_enctype == Fortezza96) ~> ciphersuite == 1);
manager.post((cipher_enctype == NoEnc || cipher_enctype == RC4128
  || cipher_enctype == AES128 || cipher_enctype == AES192
  || cipher_enctype == AES256 || cipher_enctype == RC4128
  || cipher_enctype == DE540 || cipher_enctype == DE566
  || cipher_enctype == TripleDES168 || cipher_enctype == IDEA128
  || cipher_enctype == Fortezza96));

// Cross-constraints
APPENDIX D. SSL/TLS ANALYSIS

// Authentication require the configuration of a ciphersuite
manager.post(authenticationmethod == 1 ~> ciphersuite == 1);

// Certain protocols supports specific keychangemethod and ciphers

// SSLv2
manager.post((protocol == SSLv2 && certificate == 1) ~> (keychangemethod == RSA || x509 == 1 || mac == MD5 &&
(cipher_enctype == NoEnc || cipher_enctype == RC4128 || cipher_enctype == DES192 || cipher_enctype == RC2128EX
|| cipher_enctype == DES64 || cipher_enctype == RC4128EXP || cipher_enctype == RC2128EXP));

// SSLv3
// RSA and certificate
manager.post((protocol == SSLv3 && keychangemethod == RSA) ~> (cipher_enctype == NoEnc && mac == MD5 ||
cipher_enctype == RC4128 && mac == MD5 ||
cipher_enctype == RC240 && mac == SHA1 ||
cipher_enctype == TripleDES168 && mac == SHA1)));

// DH and certificate
manager.post((protocol == SSLv3 && keychangemethod == DH,DSS) ~> (cipher_enctype == NoEnc && mac == SHA1 ||
cipher_enctype == RC4128 && mac == SHA1 ||
cipher_enctype == TripleDES168 && mac == SHA1));

manager.post((protocol == SSLv3 && keychangemethod == DH_RSA) ~> (cipher_enctype == NoEnc && Mac == SHA1 ||
cipher_enctype == TripleDES168 && Mac == SHA1));

manager.post((protocol == SSLv3 && keychangemethod == DHE_DSS) ~> (cipher_enctype == NoEnc && Mac == SHA1 ||
cipher_enctype == TripleDES168 && Mac == SHA1));

manager.post((protocol == SSLv3 && keychangemethod == DHE_RSA) ~> (cipher_enctype == NoEnc && Mac == SHA1 ||
cipher_enctype == DHE && Mac == SHA1 ||
cipher_enctype == TripleDES168 && Mac == SHA1));

// DH anonymous
manager.post((protocol == SSLv3 && keychangemethod == DH_anon) ~> (cipher_enctype == NoEnc && Mac == SHA1 ||
cipher_enctype == RC4128 && Mac == SHA1 ||
cipher_enctype == TripleDES168 && Mac == SHA1));

// Fortezza PSK or SRP
manager.post((protocol == SSLv3 && keychangemethod == Fortezza) ~> (cipher_enctype == NoEnc && Mac == SHA1 ||
cipher_enctype == RC4128 && Mac == SHA1));

// TLS
// RSA and certificate
manager.post((protocol == TLSv11 && keychangemethod == RSA) ~> (cipher_enctype == NoEnc && Mac == SHA1 ||
cipher_enctype == RC4128 && Mac == SHA1 ||
cipher_enctype == RC4128 && Mac == SHA1 ||
cipher_enctype == RC240 && Mac == SHA1 ||
cipher_enctype == TripleDES168 && Mac == SHA1));
D.1. CONSTRAINT MODEL FOR FEATURE MODEL OF SSL/TLS

manager.post((protocol == TLSv1 & keychangemethod == RSA) ~>
(cipher_enctype == NoEnc & mac == MD5 |)
cipher_enctype == NoEnc & mac == SHA1 |
cipher_enctype == RC440 & mac == MD5 |
cipher_enctype == RC4128 & mac == MD5 |
cipher_enctype == RC4128 & mac == SHA1 |
cipher_enctype == RC240 & mac == MD5 |
cipher_enctype == IDEA128 & mac == SHA1 |
cipher_enctype == DES40 & mac == SHA1 |
cipher_enctype == DES & mac == SHA1 |
cipher_enctype == TripleDES168 & mac == SHA1));
manager.post((protocol == TLSv12 & keychangemethod == RSA) ~>
(cipher_enctype == NoEnc & mac == MD5 |)
cipher_enctype == NoEnc & mac == SHA1 |
cipher_enctype == NoEnc & mac == SHA256 |
cipher_enctype == RC4128 & mac == SHA1 |
cipher_enctype == TripleDES168 & mac == SHA1 |
cipher_enctype == AES128 & mac == SHA1 |
cipher_enctype == AES256 & mac == SHA1 |
cipher_enctype == AES128 & mac == SHA256 |
cipher_enctype == AES256 & mac == SHA256));

// DH and certificates
manager.post((protocol == TLSv11 & keychangemethod == DH_DSS) ~>
(cipher_enctype == DES & mac == SHA1 |
cipher_enctype == TripleDES168 & mac == SHA1));
manager.post((protocol == TLSv11 & keychangemethod == DH_RSA) ~>
(cipher_enctype == DES & mac == SHA1 |
cipher_enctype == TripleDES168 & mac == SHA1));
manager.post((protocol == TLSv11 & keychangemethod == DHE_DSS) ~>
(cipher_enctype == DES & mac == SHA1 |
cipher_enctype == TripleDES168 & mac == SHA1));
manager.post((protocol == TLSv11 & keychangemethod == DHE_RSA) ~>
(cipher_enctype == DES & mac == SHA1 |
cipher_enctype == TripleDES168 & mac == SHA1));
manager.post((protocol == TLSv1 & keychangemethod == DH_DSS) ~>
(cipher_enctype == DES40 & mac == SHA1 |
cipher_enctype == TripleDES168 & mac == SHA1));
manager.post((protocol == TLSv1 & keychangemethod == DH_RSA) ~>
(cipher_enctype == DES40 & mac == SHA1 |
cipher_enctype == TripleDES168 & mac == SHA1));
manager.post((protocol == TLSv1 & keychangemethod == DHE_DSS) ~>
(cipher_enctype == DES40 & mac == SHA1 |
cipher_enctype == TripleDES168 & mac == SHA1));
manager.post((protocol == TLSv1 & keychangemethod == DHE_RSA) ~>
(cipher_enctype == DES40 & mac == SHA1 |
cipher_enctype == TripleDES168 & mac == SHA1));
manager.post((protocol == TLSv12 & keychangemethod == DH_DSS) ~>
(cipher_enctype == AES128 & mac == SHA1 |
cipher_enctype == AES256 & mac == SHA1 |
cipher_enctype == AES128 & mac == SHA256 |
cipher_enctype == AES256 & mac == SHA256 |
cipher_enctype == TripleDES168 & mac == SHA1));
manager.post((protocol == TLSv12 & keychangemethod == DH_RSA) ~>
(cipher_enctype == AES128 & mac == SHA1 |
cipher_enctype == AES256 & mac == SHA1 |
cipher_enctype == AES128 & mac == SHA256 |
cipher_enctype == AES256 & mac == SHA256 |
cipher_enctype == TripleDES168 & mac == SHA1));
manager.post((protocol == TLSv12 & keychangemethod == DHE_DSS) ~>
(cipher_enctype == AES128 & mac == SHA1 |
cipher_enctype == AES256 & mac == SHA1 |
cipher_enctype == AES128 & mac == SHA256 |
cipher_enctype == AES256 & mac == SHA256 |
cipher_enctype == TripleDES168 & mac == SHA1));
manager.post((protocol == TLSv12 & keychangemethod == DHE_RSA) ~>
APPENDIX D. SSL/TLS ANALYSIS

(cipher_enctype == AES128 && mac == SHA1 ||
cipher_enctype == AES256 && mac == SHA1 ||
cipher_enctype == AES128 && mac == SHA256 ||
cipher_enctype == AES256 && mac == SHA256 ||
cipher_enctype == TripleDES168 && mac == SHA1 ));

manager.post((protocol == TLSv12 && keychangemethod == DHE_RSA) ~> (cipher_enctype == AES128 && mac == SHA1 ||
cipher_enctype == AES256 && mac == SHA1 ||
cipher_enctype == AES128 && mac == SHA256 ||
cipher_enctype == AES256 && mac == SHA256 ||
cipher_enctype == TripleDES168 && mac == SHA1 ));

// DH anonymous
manager.post((protocol == TLSv11 && keychangemethod == DH_anon) ~> (cipher_enctype == RC4128 && mac == MD5 ||
cipher_enctype == DES && mac == SHA1 ||
cipher_enctype == TripleDES168 && mac == SHA1));

manager.post((protocol == TLSv1 && keychangemethod == DH_anon) ~> (cipher_enctype == RC440 && mac == MD5 ||
cipher_enctype == RC4128 && mac == MD5 ||
cipher_enctype == DES40 && mac == SHA1 ||
cipher_enctype == DES && mac == SHA1 ||
cipher_enctype == TripleDES168 && mac == SHA1));

manager.post((protocol == TLSv12 && keychangemethod == DH_anon) ~> (cipher_enctype == RC4128 && mac == MD5 ||
cipher_enctype == TripleDES168 && mac == SHA1 ||
cipher_enctype == AES128 && mac == SHA1 ||
cipher_enctype == AES256 && mac == SHA256 ||
cipher_enctype == AES256 && mac == SHA256));

// Section for attributes (ARO_reduction)
manager.post(protocol == SSLv2 ~> (ssl_tls_aroreduction_ARP >= 20 &&
ssl_tls_aroreduction_ARP <= 60));
manager.post(protocol == SSLv3 ~> ssl_tls_aroreduction_ARP >= 30 &&
ssl_tls_aroreduction_ARP <= 70));
manager.post(protocol == TLSv1 ~> ssl_tls_aroreduction_ARP >= 40 &&
ssl_tls_aroreduction_ARP <= 85));
manager.post(protocol == TLSv11 ~> ssl_tls_aroreduction_ARP >= 40 &&
ssl_tls_aroreduction_ARP <= 90));
manager.post(protocol == TLSv12 ~> ssl_tls_aroreduction_ARP >= 40 &&
ssl_tls_aroreduction_ARP <= 95));

manager.post(x509 == 1 ~> (ssl_tls_aroreduction_certificate_x509 >= 40 &&
ssl_tls_aroreduction_certificate_x509 <= 75));
manager.post(x509 == 0 ~> (ssl_tls_aroreduction_certificate_x509 == 0));
manager.post(OpenPGP == 1 ~> (ssl_tls_aroreduction_certificate_OpenPGP >= 30 &&
ssl_tls_aroreduction_certificate_OpenPGP <= 70));
manager.post(OpenPGP == 0 ~> (ssl_tls_aroreduction_certificate_OpenPGP == 0));
manager.post(certificate == 1 ~> ssl_tls_aroreduction_certificate == (ssl_tls_aroreduction_certificate_x509 +
ssl_tls_aroreduction_certificate_OpenPGP));
manager.post(certificate == 0 ~> ssl_tls_aroreduction_certificate == 0);
manager.post(digital_signature == 1 ~> ssl_tls_aroreduction_certificate == 0);
manager.post(ssl_tls == 1 ~> (ssl_tls_aroreduction ==
((ssl_tls_aroreduction_ARP + ssl_tls_aroreduction_certificate)/2)));
D.1. CONSTRAINT MODEL FOR FEATURE MODEL OF SSL/TLS

```plaintext
manager.post(ssl_tls == 0 ~> (ssl_tls_ale == 0));
manager.post(ssl_tls == 0 ~> (ssl_tls_aroreduction == 0));

// manager.post(ssl_tls == 1 ~> (ssl_tls_aroreduction >= 80 
&& ssl_tls_aroreduction <= 80));
manager.post(ssl_tls == 1 ~> (ssl_tls_ale >= 2000 
&& ssl_tls_ale <= 2000));

// Section for attributes (Cost)
manager.post(protocol == SSLv2 ~> (ssl_tls_protoclo_cost >= 10 && 
ssl_tls_protoclo_cost <= 50));
manager.post(protocol == SSLv3 ~> ssl_tls_protoclo_cost >= 20 && 
ssl_tls_protoclo_cost <= 55));
manager.post(protocol == TLSv1 ~> ssl_tls_protoclo_cost >= 30 && 
ssl_tls_protoclo_cost <= 60));
manager.post(protocol == TLSv11 ~> ssl_tls_protoclo_cost >= 35 && 
ssl_tls_protoclo_cost <= 65));
manager.post(protocol == TLSv12 ~> ssl_tls_protoclo_cost >= 40 && 
ssl_tls_protoclo_cost <= 70));
manager.post(x509 == 1 ~> (ssl_tls_certificate_x509_cost >= 15 
&& ssl_tls_certificate_x509_cost <= 50) 
&& ssl_tls_certificate_OpenPGP_cost == 0 
&& ssl_digitalsignature_cost == 0);
manager.post(x509 == 0 ~> (ssl_tls_certificate_x509_cost == 0));
manager.post(OpenPGP == 1 ~> (ssl_tls_certificate_OpenPGP_cost >= 25 
&& ssl_tls_certificate_OpenPGP_cost <= 40) 
&& ssl_tls_certificate_x509_cost == 0 
&& ssl_digitalsignature_cost == 0 ));
manager.post(OpenPGP == 0 ~> (ssl_tls_certificate_OpenPGP_cost == 0));
manager.post(certificate == 1 ~> (ssl_tls_certificate_cost == 
ssl_tls_certificate_OpenPGP_cost + ssl_tls_certificate_x509_cost));
manager.post(certificate == 0 ~> (ssl_tls_certificate_cost == 0));
manager.post(digital_signature == 1 ~> 
ssl_digitalsignature_cost >= 10 && ssl_digitalsignature_cost <= 30 
&& ssl_tls_certificate_cost == 0));
manager.post(digital_signature == 0 ~> ssl_digitalsignature_cost == 0);
manager.post(ssl_tls == 1 ~> (ssl_tls_cost == (ssl_tls_protoclo_cost + 
ssl_tls_certificate_cost + ssl_digitalsignature_cost)));
manager.post(ssl_tls == 0 ~> (ssl_tls_cost == 0));
manager.post(ssl_tls == 1 ~> (ssl_tls_cost >= 50 && ssl_tls_cost <= 50));
}

using{
label(secure_protocol);
label(integrity);
label(confidentiality);
label(authentication);
label(ssl_tls);
label(protocol);
label(ciphersuite);
label(authenticationmethod);
label(keychangemethod);
label(mac);
label(cipher_enctype);
label(certificate);
label(digital_signature);
label(x509);
label(OpenPGP);
```
label(SRP);
label(PSK);
label(Anonymous);

label(ssl_tls_aroreduction);
label(ssl_tls_aroreduction_ARP);
label(ssl_tls_aroreduction_certificate);
label(ssl_tls_aroreduction_certificate_x509);
//label(ssl_tls_sle);
//label(ssl_tls_aro);
//label(ssl_tls_openpgp);
label(ssl_tls_cost);
label(ssl_tls_protocol_cost);
label(ssl_tls_certificate_cost);
label(ssl_digitsignature_cost);
label(ssl_tls_certificate_x509_cost);
label(ssl_tls_certificate_OpenPGP_cost);

outfile << "SecureProtocol:" << secure_protocol<< endl;
outfile << "Confidentiality:" << confidentiality<< endl;
outfile << "Integrity:" << integrity<< endl;
outfile << "Authentication:" << authentication<< endl;
outfile << "SSLTLS:" << ssl_tls<< endl;
outfile << "SessionKey:" << sessionkey<< endl;
outfile << "Protocol:" << protocol<< endl;
outfile << "CipherSuite:" << ciphersuite<< endl;
outfile << "AuthenticationMethod:" << authenticationmethod<< endl;
outfile << "KeyChangeMethod:" << keychangemethod<< endl;
outfile << "MAC:" << mac<< endl;
outfile << "CipherEnc:" << cipher_encrypttype<< endl;
outfile << "Certificate:" << certificate<< endl;
outfile << "DigitalSignature:" << digital_signature<< endl;
outfile << "OpenPGP:" << OpenPGP<< endl;
outfile << "SRP:" << SRP<< endl;
outfile << "PSK:" << PSK<< endl;
outfile << "Anonymous:" << Anonymous<< endl;

outfile << "ARO:" << ssl_tls_aroreduction<< endl;
outfile << "AROR:" << ssl_tls_aroreduction_ARP<< endl;
outfile << "ARORC:" << ssl_tls_aroreduction_certificate<< endl;
outfile << "ARORx509:" << ssl_tls_aroreduction_certificate_x509<< endl;
outfile << "AROROGP:" << ssl_tls_aroreduction_certificate_OpenPGP<< endl;
outfile << "SLE:" << ssl_tls_sle<< endl;
outfile << "PC:" << ssl_tls_protocol_cost<< endl;
outfile << "CC:" << ssl_tls_certificate_cost<< endl;
outfile << "DC:" << ssl_digitsignature_cost<< endl;
outfile << "s509C:" << ssl_tls_certificate_s509_cost<< endl;
outfile << "OpenPGPC:" << ssl_tls_certificate_OpenPGP_cost<< endl;
outfile << "CostTotal:" << ssl_tls_cost<< endl;

}  

/********** Print out section **********/  

cout << "#Search info. : " << manager.getStore()<< endl;
cout << "#fails:" << manager.getNFail() << endl;
cout << "#solutions:" << solutions.getSize() << endl;
cout << "#1SLtime: " << System.getWCTime() - startTime << " ms" << endl;
Appendix E

Feature Analysis of Web Application Firewall (WAF) Systems

In Chapter 5, we have presented a feature model diagram drawn in Figure 5.10. The code in below represents the formal constraint model of this feature model. The code also includes metrics and extra functionalities defined in the chapter. This model together with previous ones are available for downloading in (Varela-Vaca 2012).

WAFs are stated by means of rules. One rule specified, for instance, by ModSecurity and IronBee has the following structure: Rule VARIABLES OPERATOR [ACTIONS]. Furthermore, rules could specify a set of operations (Operations/Transformation) used to inspect the content. In the tables below, the different possibilities that can be used in the definition of a rule are described. The analysis is focused on ModSecurity, Ironbee, and ESAPIWAF systems.

Constraint Model for Feature Model of WAFs

```java
import cotfd;

/**************** Configuration **************/
Solver<CP> manager(); // Solver manager
SolutionPool solutions(); // Pool of solutions
ofstream outfile("d:\outfile_CA.txt");

int startTime = System.getWCTime(); // Set up begining time
int threshold = 3000;

AbstractSearchController searchController = manager.getSearchController();

// Solutions are checked if feasible
whenever searchController.onFeasibleSolution(Solution s){
    // solutions is stored in the pool of solutions
    solutions.add(s);
}

/**************** Types **************/
enum Variables = {ARGS,PATH_INFO,RESPONSE,REQUEST,REMOTE_ADDR,AUTH_TYPE,FILES,ENV};
enum DisruptiveActions = {allow,block,deny,drop,pass,pause,phase,proxy,redirect,status};
```
enum MetadataActions = {id, logdata, msg, rev, severity};
enum NonDisruptiveActions =
{append, auditlog, capture, cti, deprecatevar, exec, expirevar, initcti, log, multimatch, noauditlog, nolog, setuid, setsid, setenv, setvar, skip, sanitisearg, prepend, tag, xmlname};
enum FlowActions = {chain};
enum Operations = {beginsWith, contains, endsWith, eq, ge, geoLookup, gsbLookup, gt, inspectFile, ipMatch, le, lt, pm, pmf, pmFromFile, rbl, rsub, rx, streq, strmatch, validateByteRange, validateDTD, validateSchema, validateUrlEncoding, validateUtf8Encoding, verifyCC, verifyCPF, verifySSR, within};

/******* Ranges/Variables/Constraints CP ***********/

range literal = 0..1;

// Web Application Firewall
var<CP>{int} waf(manager, literal);
var<CP>{int} rule(manager, literal);
var<CP>{int} phase(manager, literal);
var<CP>{int} phase1(manager, literal);
var<CP>{int} phase2(manager, literal);
var<CP>{int} phase3(manager, literal);
var<CP>{int} phase4(manager, literal);
var<CP>{int} phase5(manager, literal);

// Rules might be defined by a set of variables (at least one)
var<CP>{int} variable(manager, Variables);
var<CP>{int} expression(manager, literal);

// Rules might contain a set of actions
// (optional for each rule, there exist a default action)
var<CP>{int} disruptiveActions(manager, DisruptiveActions);
var<CP>{int} nondisruptiveActions(manager, NonDisruptiveActions);
var<CP>{int} metadataActions(manager, MetadataActions);
var<CP>{int} flowActions(manager, FlowActions);

// Rules could contain transformations operations (optional)
var<CP>{int} transformation(manager, literal);

// A rule can contain a set of operators to be applied (optional)
var<CP>{int} operation(manager, Operations);

/******** Solver ***********/

Integer np(manager.getNPropag());
cout << "Solving feature model: " << endl;
manager.limitTime(20);
solveall<manager> {

// Rule
manager.post(waf == 1 ~> rule == 1);
manager.post(rule == 1 ~> waf == 1);

// Transformation
manager.post(transformation == 1 ~> waf == 1);

// Operation
manager.post((operation == beginsWith || operation == contains || operation == eq) || (operation == endsWith || operation == beginSS || operation == endsSS || operation == eq)) || (operation == geoLookup || operation == gsbLookup || operation == gt) || (operation == inspectFile || operation == ipMatch || operation == ge) || (operation == le) || (operation == lt) || (operation == pm) || (operation == pmf)
|| operation == pmFromFile|| operation == rbl|| operation == rsrb
|| operation == rx|| operation == streql|| operation == strmatch
|| operation == validateByteRange|| operation == validateDTD
|| operation == validateSchema|| operation == validateUrlEncoding
|| operation == validateUtf8Encoding|| operation == verifyCC
|| operation == verifyCPF|| operation == verifySSN
|| operation == within) ~> waf == 1);

//Phase
manager.post(waf == 1 ~> (variable == ARGS || variable == PATH_INFO
|| variable == RESPONSE|| variable == REQUEST|| variable == REMOTE_ADDR
|| variable == AUTH_TYPE || variable == FILES || variable == ENV));
manager.post((variable == ARGS || variable == PATH_INFO
|| variable == RESPONSE|| variable == REQUEST|| variable == REMOTE_ADDR
|| variable == AUTH_TYPE || variable == FILES || variable == ENV)
~> waf == 1);
manager.post(transformation == 1 ~> waf == 1);
// each rule can use an action itself but at least one by default
manager.post(waf == 1 ~> action == 1);
manager.post(action == 1 ~> waf == 1);
manager.post((disruptiveActions == allow || disruptiveActions == block
|| disruptiveActions == deny|| disruptiveActions == drop
|| disruptiveActions == pass || disruptiveActions == pause
|| disruptiveActions == phase || disruptiveActions == proxy
|| disruptiveActions == redirect || disruptiveActions == status) ~> waf == 1);
manager.post(waf == 1 ~> (disruptiveActions == allow
|| disruptiveActions == block|| disruptiveActions == deny
|| disruptiveActions == drop || disruptiveActions == pass
|| disruptiveActions == pause || disruptiveActions == phase
|| disruptiveActions == proxy || disruptiveActions == redirect
|| disruptiveActions == status));
manager.post((nondisruptiveActions == append
|| nondisruptiveActions == auditlog|| nondisruptiveActions == capture
|| nondisruptiveActions == cti|| nondisruptiveActions == deprecatedvar
|| nondisruptiveActions == exec|| nondisruptiveActions == expirevar
|| nondisruptiveActions == initctil|| nondisruptiveActions == log
|| nondisruptiveActions == multmatcher| nondisruptiveActions == noauditlog
|| nondisruptiveActions == nolog|| nondisruptiveActions == setuid
|| nondisruptiveActions == setsid|| nondisruptiveActions == setenv
|| nondisruptiveActions == setvar|| nondisruptiveActions == skip
|| nondisruptiveActions == sanitisearg|| nondisruptiveActions == prependt
|| nondisruptiveActions == tag|| nondisruptiveActions == xmlns) ~> waf == 1);
manager.post((metadataActions == id || metadataActions == logdata
|| metadataActions == msg|| metadataActions == rev
|| metadataActions == severity) ~> waf == 1);
manager.post((flowActions == chain) ~> waf == 1);

//Phase
manager.post(phase == 1 ~> waf == 1);
manager.post(waf == 1 ~> phase == 1);
manager.post((phase == 1) ~> (phase1 == 1 || phase2 == 1
|| phase3 == 1 || phase4 == 1 || phase5 == 1) &&
!(phase1 == 1 && phase2 == 0 && phase3 == 0
&& phase4 == 0 && phase5 == 0) &&
!(phase1 == 1 && phase2 == 0 && phase3 == 1
&& phase4 == 0 && phase5 == 0) &&


APPENDIX E. WAFS ANALYSIS

!(phase1 == 1 && phase2 == 0 && phase3 == 0 && phase4 == 1 && phase5 == 0) &&
!(phase1 == 1 && phase2 == 0 && phase3 == 0 && phase4 == 0 && phase5 == 1) &&
!(phase1 == 1 && phase2 == 1 && phase3 == 1 && phase4 == 0 && phase5 == 0) &&
!(phase1 == 1 && phase2 == 1 && phase3 == 0 && phase4 == 1 && phase5 == 1) &&
!(phase1 == 1 && phase2 == 1 && phase3 == 1 && phase4 == 1 && phase5 == 0) &&
!(phase1 == 1 && phase2 == 1 && phase3 == 0 && phase4 == 1 && phase5 == 1) &&
!(phase1 == 0 && phase2 == 1 && phase3 == 1 && phase4 == 0 && phase5 == 0) &&
!(phase1 == 0 && phase2 == 1 && phase3 == 0 && phase4 == 1 && phase5 == 0) &&
!(phase1 == 0 && phase2 == 1 && phase3 == 0 && phase4 == 0 && phase5 == 1) &&
!(phase1 == 0 && phase2 == 1 && phase3 == 1 && phase4 == 1 && phase5 == 0) &&
!(phase1 == 0 && phase2 == 1 && phase3 == 1 && phase4 == 1 && phase5 == 1) &&
!(phase1 == 0 && phase2 == 0 && phase3 == 1 && phase4 == 1 && phase5 == 0) &&
!(phase1 == 0 && phase2 == 0 && phase3 == 1 && phase4 == 1 && phase5 == 1) &&
!(phase1 == 0 && phase2 == 0 && phase3 == 0 && phase4 == 1 && phase5 == 1);
&& phase4 == 1 && phase5 == 0) &&
!(phase1 == 0 && phase2 == 0 && phase3 == 1 && phase4 == 1 && phase5 == 1) &&
!(phase1 == 0 && phase2 == 0 && phase3 == 0 && phase4 == 1 && phase5 == 1));
}

using{
label(waf);
label(rule);
label(variable);
label(transformation);
label(phase);
label(phase1);
label(phase2);
label(phase3);
label(phase4);
label(phase5);
label(action);
label(disruptiveActions);
label(nondisruptiveActions);
label(metadataActions);
label(flowActions);
// label(operation);
label(transformation);

//outfile << "****** Solution ******" << endl;
outfile << "waf:" << waf << " ";
outfile << "rule:" << rule << " ";
outfile << "variable:" << variable << " ";
outfile << "phase:" << phase << " ";
outfile << "phase1:" << phase1 << " ";
outfile << "phase2:" << phase2 << " ";
outfile << "phase3:" << phase3 << " ";
outfile << "phase4:" << phase4 << " ";
outfile << "phase5:" << phase5 << " ";
outfile << "transformation:" << transformation << " ";
outfile << "action:" << action << " ";
outfile << "disruptiveActions:" << disruptiveActions << " ";
outfile << "nondisruptive:" << nondisruptiveActions << " ";
outfile << "metadataActions:" << metadataActions << " ";
outfile << "flowActions:" << flowActions << " ";
outfile << "transformation:" << transformation << " ";
outfile << "operation:" << operation << endl;
//outfile << "*****************" << endl;

/********* Print out section ***********/
cout << "#Search info. : " << manager.getStore()<< endl;
cout << "#fails:" << manager.getNFail() << endl;
cout << "#solutions:" << solutions.getSize() << endl;
cout << "#1SLtime: " << System.getWCTime() - startTime << " ms" << endl;
Appendix F

Feature Model for Fault Tolerance

Code in below represents the COMET constraint model for obtaining configurations of fault tolerance such as described in Table 5.3 and drawn in Figure 5.6. This model together previous ones are available for downloading in [Varela-Vaca, 2012].

```cpp
import cotfd;

/********** Configuration **********/
Solver<CP> manager(); // Solver manager
SolutionPool solutions(); // Pool of solutions
ofstream outfile("d:\outfile_ft.txt");

int startTime = System.getWCTime(); // Set up begining time
int threshold = 3000;

AbstractSearchController searchController = manager.getSearchController();

// Solutions are checked if feasiable
whenever searchController@onFeasibleSolution(Solution s){
    // solutions is stored in the pool of solutions
    // cout << "Objetivo ajustado" << endl;
    solutions.add(s);
}

enum adjudicator = {enull,exact_majority,median,mean,consensus};

/********** Ranges/Variables/Constraints CP **********/

range literal = 0..1;

// Fault Tolerance
var<CP>(int) ft(manager,1..1);

// Features
var<CP>(int) db(manager,literal);
var<CP>(int) nor(manager,literal);
var<CP>(int) dboracle(manager,literal);
var<CP>(int) dbdiagnoser(manager,literal);
var<CP>(int) dbrules(manager,literal);
var<CP>(int) binder(manager,literal);
var<CP>(int) bbinder(manager,literal);
var<CP>(int) nvp(manager,literal);
var<CP>(int) nov(manager,literal);
var<CP>(int) voter(manager,adjudicator);
```

209
APPENDIX F. FEATURE MODEL FOR FAULT TOLERANCE

var<CP>{int} cp(manager,literal);
var<CP>{int} nos(manager,literal);
var<CP>{int} cporacle(manager,literal);
var<CP>{int} cpdiagnoser(manager,literal);
var<CP>{int} cprules(manager,literal);
var<CP>{int} ch(manager,literal);

// Attributes: Time and MTTF
range time_range = 1..10000;
var<CP>{int} ft_mttf(manager,time_range);
var<CP>{int} nvp_time(manager,0..500);
var<CP>{int} db_time(manager,0..500);
var<CP>{int} cp_time(manager,0..500);
range oracle_time_range = 0..20;
var<CP>{int} dboracle_time(manager,oracle_time_range);
var<CP>{int} cporacle_time(manager,oracle_time_range);
var<CP>{int} dbdiagnoser_time(manager,oracle_time_range);
var<CP>{int} cpdiagnoser_time(manager,oracle_time_range);
var<CP>{int} dbrules_time(manager,oracle_time_range);
var<CP>{int} cprules_time(manager,oracle_time_range);
range nor_time_range = 0..20;
var<CP>{int} nor_time(manager,nor_time_range);
range binder_time_range = 0..40;
var<CP>{int} binder_time(manager,binder_time_range);
var<CP>{int} bbinder_time(manager,binder_time_range);
range nov_time_range = 0..20;
var<CP>{int} nov_time(manager,nov_time_range);
var<CP>{int} adj_time(manager,0..500);
var<CP>{int} EM_time(manager,0..30);
var<CP>{int} M_time(manager,0..40);
var<CP>{int} m_time(manager,0..50);
var<CP>{int} c_time(manager,0..60);
range nos_time_range = 0..20;
var<CP>{int} nos_time(manager,nos_time_range);
var<CP>{int} ch_time(manager,0..50);

// Attributes: Risk Reduction
range rr_range = 0..100;
var<CP>{int} ft_rr(manager,rr_range);
var<CP>{int} nvp_rr(manager,rr_range);
var<CP>{int} db_rr(manager,rr_range);
var<CP>{int} cp_rr(manager,rr_range);
var<CP>{int} nor_rr(manager,rr_range);
var<CP>{int} nos_rr(manager,rr_range);
var<CP>{int} nov_rr(manager,rr_range);
var<CP>{int} binder_rr(manager,rr_range);
var<CP>{int} bbinder_rr(manager,rr_range);
var<CP>{int} adj_rr(manager,rr_range);
var<CP>{int} EM_rr(manager,rr_range);
var<CP>{int} M_rr(manager,rr_range);
var<CP>{int} m_rr(manager, rr_range);
var<CP>{int} C_rr(manager, rr_range);

range num_replicas = 0 .. 10;
range num_replicas_aux = 1 .. 10;
var<CP>{int} nor_nr(manager, num_replicas);
var<CP>{int} nos_nr(manager, 0..2);
var<CP>{int} nov_nr(manager, 0..20);
var<CP>{int} nov_nr_aux(manager, num_replicas_aux);

// Weighted
var<CP>{int} rf1(manager, literal);
//var<CP>{int} w1(manager, 5..5);
var<CP>{int} rf2(manager, literal);
var<CP>{int} rf3(manager, 0..500);

/********* Solver ***********/
Integer np(manager.getNPropag());
cout << "Solving feature model: " << endl;
manager.limitTime(60);
// Objective function:
// minimize<manager>
// maximize<manager>
// ft_cost
// ft_mttf
// ft_rr
// nov_nr
// rf3
//subject to{
solveall<manager> {

// Feature model constraints
manager.post(ft == 1 ~> (db == 1 || nvp == 1 || cp == 1)) &&
!(db == 1 && nvp == 0 && cp == 1) &&
!(db == 0 && nvp == 1 && cp == 1) &&
!(db == 1 && nvp == 1 && cp == 0) &&
!(db == 1 && nvp == 1 && cp == 1) &&

};
manager.post(
( (db == 1 || nvp == 1 || cp == 1) ~> ft == 1 &&
!(db == 1 && nvp == 0 && cp == 1) &&
!(db == 0 && nvp == 1 && cp == 1) &&
!(db == 1 && nvp == 1 && cp == 0) &&
!(db == 1 && nvp == 1 && cp == 1) &&

});
manager.post(db == 1 ~> nor == 1);
manager.post(nor == 1 ~> db == 1);
manager.post(db == 1 ~> dboracle == 1);
manager.post(dboracle == 1 ~> db == 1);
manager.post(dboracle == 1 ~> dbdiagnoser == 1);
manager.post(dbdiagnoser == 1 ~> dboracle == 1);
manager.post(dboracle == 1 ~> dbrules == 1);
manager.post(dbrules == 1 ~> dboracle == 1);
manager.post(db == 1 ~> binder == 1);
manager.post(binder == 1 ~> db == 1);

manager.post(binder == 1 ~> dbdiagnoser == 1);
manager.post(bbinder == 1 ~> dbdiagnoser == 1);
manager.post(bbinder == 1 ~> db == 1);
manager.post(nvp == 1 ~> nov == 1);
manager.post(nov == 1 ~> nvp == 1);
manager.post(nvp == 1 ~> (voter == exact_majority ||
voter == median ||
voter == mean ||
voter == consensus) ~> nov == 1);

manager.post(cp == 1 ~> nos == 1);
manager.post(nos == 1 ~> cp == 1);
manager.post(nos == 1 ~> cpdiagnoser == 1);
manager.post(cp == 1 ~> cporacle == 1);
manager.post(cporacle == 1 ~> cp == 1);
manager.post(cporacle == 1 ~> cpdiagnoser == 1);
manager.post(cprules == 1 ~> cpdiagnoser == 1);
manager.post(cprules == 1 ~> cporacle == 1);
manager.post(cporacle == 1 ~> cprules == 1);

manager.post(cp == 1 ~> ch == 1);
manager.post(ch == 1 ~> cp == 1);

// Extra-functionalities of feature model

// Time and MTTR
// DB
manager.post((db == 1) ~> (nor_time >= 10 && nor_time <= 40) &&
(binder_time >= 10 && binder_time <= 30) &&
(dboracle_time >= 20) && (dbdiagnoser_time >= 10 &&
(dbrules_time >= 20) && (db_time == nor_time + binder_time +
(dboracle_time + dbdiagnoser_time + dbrules_time)));
manager.post((db == 1) ~> (db_time == nor_time + binder_time +
binder_time + dbdiagnoser_time + dbrules_time));
manager.post((db == 0) ~> (nor_time == 0 && binder_time == 0 &&
(bdrules_time == 0 && db_time == 0 &&
dbdiagnoser_time == 0 &&
dbrules_time == 0));

// CF
manager.post((cp == 1) ~> (nos_time >= 10 && nos_time <= 40) &&
(ch_time >= 10 && ch_time <= 30) && (cporacle_time >= 10 &&
(cporacle_time >= 20) && (cpdiagnoser_time >= 10 &&
(cprules_time >= 10 && cprules_time <= 20));
manager.post((cp == 1) ~> (cp_time == nos_time + ch_time +
cporacle_time + cpdiagnoser_time + cprules_time));
manager.post((cp == 0) ~> (nos_time == 0 && ch_time == 0 &&
cporacle_time == 0 && cp_time == 0 && cpdiagnoser_time == 0 &&
cprules_time == 0 && cprules_time <= 20));
cprules_time == 0));

//NVP
manager.post((nvp == 1) ~> (nov_time >= 10 && nov_time <= 30));
manager.post((nvp == 1 && voter == enull) ~> (adj_time >= 10 &&
adj_time <= 30));
manager.post((nvp == 1 && voter == exact_majority) ~> (adj_time >= 10 &&
adj_time <= 30));
manager.post((nvp == 1 && voter == median) ~> (adj_time >= 10 &&
adj_time <= 30));
manager.post((nvp == 1 && voter == mean) ~> (adj_time >= 10 &&
adj_time <= 30));
manager.post((nvp == 1 && voter == consensus) ~> (adj_time >= 10 &&
adj_time <= 30));
manager.post((nvp == 1) ~> (nvp_time == nov_time + adj_time));
manager.post((nvp == 0) ~> (nvp_time == 0 && adj_time == 0 && nvp_time == 0));

//FT
manager.post((ft == 1) ~> (ft_mttf == db_time + nvp_time + cp_time));
manager.post((ft == 1) ~> (ft_mttf >= 20 && ft_mttf <= 50));

// Risk Reduction
// DB
manager.post((nor == 1) ~> (nor_nr >= 1 ~> (db_rr >= 50)));
manager.post((nor == 1) ~> (bbinder == 1 ~> (db_rr >= 70)));
manager.post((nor == 1) ~> (nor_nr >= 1 && nor_nr <= 10) &&
nos_nr == 0) && (nov_nr == 0));
manager.post((nor == 0) ~> (nor_nr == 0));

// CP
manager.post((nos == 1) ~> (nos_nr >= 1 ~> (cp_rr >= 50)));
manager.post((nos == 1) ~> (nos_nr >= 1 && nos_nr <= 2) &&
nos_nr == 0) && (nos_nr == 0));
manager.post((nos == 0) ~> (nos_nr == 0));

//NVP
manager.post((nov == 1) ~> (nov_nr >= 3 && nov_nr >= 6) ~> (nvp_rr >= 60));
manager.post((nov == 1) ~> (nov_nr >= 7 && nov_nr >= 9) ~> (nvp_rr >= 80));
manager.post((nov == 1) ~> (nov_nr >= 10 ~> (nvp_rr >= 80));
manager.post((nov == 1) ~> (nov_nr == 2 + nov_nr_aux + 1) &&
nos_nr == 0) && (nos_nr == 0));
manager.post((nov == 0) ~> (nov_nr == 0));

//FT
manager.post((ft == 1) ~> (ft_rr == db_rr + nvp_rr + cp_rr));
manager.post((ft == 1) ~> (db_rr >= 30 && db_rr <= 80) &&
nos_rr == 0) && (cp_rr == 0));
manager.post((nvp == 1) ~> (nvp_rr >= 30 && nvp_rr <= 80) &&
(db_rr == 0) && (cp_rr == 0));
manager.post((cp == 1) ~> (cp_rr >= 30 && cp_rr <= 80) &&
(nvp_rr == 0) && (db_rr == 0));
manager.post(//rf2 == /*(ft_rr >= 30 && ft_rr <= 70)*/);

//manager.post(rf3 == ft_mttf + ft_rr);

// manager.post(rf3 >= 100 && rf3 <= 100);
//manager.post(rf3 >= 15 && rf3 <= 15);

}

using{
// Feature model
label(ft);
// DB
label(db);
label(nor);
label(dboracle);
label(dbrules);
label(dbdiagnoser);
label(binder);
label(bbinder);
// CP
label(cp);
label(nos);
label(cporacle);
label(cprules);
label(cpdiagnoser);
label(ch);
// HVP
label(nvp);
label(nov);
label(voter);

// Attribute

// Time and MTTR
label(ft_mttf);
label(cp_time);
label(db_time);
label(nvp_time);
label(nov_time);
label(adj_time);
label(nor_time);
label(binder_time);
label(bbinder_time);
label(cporacle_time);
label(dboracle_time);
label(cpdiagnoser_time);
label(dbdiagnoser_time);
label(cprules_time);
label(dbrules_time);
label(nos_time);
label(ch_time);

label(ft_rr);
label(db_rr);
label(nvp_rr);
label(cp_rr);

label(nor_nr);
label(nos_nr);
label(nov_nr);

//label(rf1);
//label(rf2);
// label(rf3);
outfile << "****** Solution ******" << endl;
outfile << "ft:" << ft << " ";
outfile << "db:" << db << " ";
outfile << "nor:" << nor << " ";
outfile << "dboracle:" << dboracle << " ";
outfile << "dbrules:" << dbrules << " ";
outfile << "dbdiagnoser:" << dbdiagnoser << " ";
outfile << "binder:" << binder << " ";
outfile << "bbinder:" << bbinder << " ";
outfile << "cp:" << cp << " ";
outfile << "nos:" << nos << " ";
outfile << "cporacle:" << cporacle << " ";
outfile << "cprules:" << cprules << " ";
outfile << "cpdiagnoser:" << cpdiagnoser << " ";
outfile << "ch:" << ch << " ";
outfile << "ch_time:" << ch_time << " ";
outfile << "voter:" << voter << " ";
outfile << "voter_time:" << voter_time << " ";
}
/******* Print out section ***********/
cout << "#Search info. : " << manager.getStore() << endl;
cout << "#fails:" << manager.getNFail() << endl;
cout << "#choice:" << manager.getNChoice() << endl;
cout << "#solutions:" << solutions.getSize() << endl;
cout << "#solutions_time: " << System.getWCTime() - startTime << " ms" << endl;
Bibliography


J. Cassar and M. Staroswiecki. A structural approach for the design of failure detection and identification systems. IFAC-IFIP-IMACS Conf. on Control of Industrial Processes, Belfort, France, 1997.


Gobierno de España. Real decreto 3/2010, de 8 de enero, por el que se regula el esquema nacional de seguridad en el ámbito de la administración electrónica, 2010.


BIBLIOGRAPHY


W. van der Aalst. Formalization and verification of event-driven process chains, 1999.


A. J. Varela-Vaca and R. M. Gasca. Propuesta para la generación y selección adaptable de configuraciones de seguridad para sistemas de gestión de procesos de negocio, 2012.


