Except as otherwise permitted under the Copyright, Designs and Patents Act 1988, this thesis may only be produced, stored or transmitted in any form or by any means with the prior permission in writing of the author. The author asserts his/her right to be identified as such in accordance with the terms of the Copyright, Designs and Patents Act 1988.
Policy-based Runtime Tracking
for E-learning Environments

PhD Thesis
Turki Mohammed Alghamdi

This thesis is submitted in partial fulfillment of the requirements for the Doctor of Philosophy

Software Technology Research Laboratory
Faculty of Technology
De Montfort University

January 2013
Declaration of Authorship

I, Turki Alghamdi, declare that this thesis entitled Policy-based Runtime Tracking for E-learning Environments and the work presented in it are my own and original. It is submitted for the degree of Doctor of Philosophy at De Montfort University. The work was undertaken between January 2009 and September 2012.
To my dear parents and my beloved wife..
Abstract

The Virtual Learning Environment (VLE) is a form of e-learning environment that is becoming widely adopted in higher educational institutions and universities. The term “Tracking” in relation to an e-learning context is the learner’s observation process of any possible interaction with learning activities. Learning activities are a collection of objects designed within e-learning environments to support learners in better understanding and fulfilling the learning objectives during the learning process. A tracking tool is an essential tool fixed within e-learning environments. Most current VLEs (e.g. MOODLE and Blackboard) have utilized similar tracking functions, which aim at recording statistical data for each learner. The current e-learning environments are unable to track individual learning activities where the tracked information can be used to support and guide learners. In this thesis, we propose a policy-based runtime tracking system. Such a tracking system is implemented as an integral part of an e-learning environment (e.g. MOODLE). Our proposed approach does the following: 1) keeps track of and captures the learning activity events and learner interaction events within a learning activity; 2) enforces a set of policies at runtime that specify how to manage the learning activities and the way the learners behave during them; and 3) provides the learners with supportive feedback in a timely manner.

We present a computational model which defines the behaviour of the system’s components and describes the tracking mechanism applied in our proposed approach. We designed architecture for our proposed approach with respect to the computational model. We present learning activity policies based on the ECA model to be enforced at runtime; this is done in response to the captured events about either the learning activities or the interactions of learners within the learning activities. A policy-based enforcement mechanism is proposed where learning activity policies are specified to support and guide learners to achieve the learning objective and thus to meet such requirements. Finally, we present a case study based on a current e-learning environment to evaluate our approach.
Acknowledgements

In the name of Allah, the Most Merciful and the Most Gracious, I give praise and thanks to Him for supporting me with the strength to complete this thesis, and for providing me with knowledgeable and caring individuals during the study process. This thesis could not have been completed without the recommendations, advice and encouragements of many people. It may not be possible to mention all of them here, but their contributions, guidance and support are extremely appreciated.

I would like to thank both Prof. Hussein Zedan, who was my first supervisor, he was generous with me, provided freely of his knowledge and made many suggestions, and Dr. Antonio Cau who was my second supervisor and has encouraged my autonomy and given me invaluable instruction.

I also acknowledge those individually with whom we discussed issues that are addressed in this thesis, including Dr. Helge Janicke, Dr. Amelia Platt and Dr. Francois Siewe.

Dr. Ali Alzahrani is more than a friend. Without his help, encouragement, and academic discussion, the pace of my work would be slower. I will never forget the nice times we spent working together at the university.

My final and greatest debt is to my family; my dear parents, my beloved wife, my dear child (Juri), my brothers and my sisters. The list of sacrifice they have made in order for me to complete this study is endless. Their support and love have been wonderful.
Publications


# Contents

Declaration of Authorship ...................................................... i

Abstract .................................................................................. iii

Acknowledgements ................................................................... iv

Publications .............................................................................. v

List of Figures .......................................................................... x

List of Tables ............................................................................ xii

Abbreviations ............................................................................ xiii

## 1 Introduction

1.1 Background ........................................................................ 1

1.2 Motivation ........................................................................ 3

1.3 Research Questions .......................................................... 4

1.4 Original Contributions ...................................................... 4

1.5 Research Methodology ....................................................... 5

1.6 Criteria of Success ............................................................ 9

1.7 Thesis Outline ................................................................... 9

## 2 Research Background

2.1 Introduction ........................................................................ 13

2.2 Overview on Learning Methods .......................................... 13

2.2.1 Traditional Learning (Classroom-Based) ....................... 13

2.2.2 E-learning ..................................................................... 15

2.2.2.1 Types of E-learning ...................................................... 16

2.2.3 Blended Learning .......................................................... 17

2.3 E-learning Environments ................................................... 18

2.3.1 Virtual Learning Environment (VLE) .......................... 19
2.4 Learning Activities in E-learning .................................................. 23
  2.4.1 Assessment ................................................................. 24
  2.4.2 Communication and Collaboration ................................. 25
  2.4.3 Contents-Interaction .................................................. 27
2.5 Tracking Process ................................................................. 27
  2.5.1 Tracking in Traditional Learning ................................. 27
  2.5.2 Tracking in E-learning ............................................... 28
  2.5.3 Review on Existing Tracking Tools .............................. 30
    2.5.3.1 E-Learning Monitoring Agent (ELMA) Tracking Tool .................................................. 30
    2.5.3.2 AdeLE Tracking Tool ........................................... 31
    2.5.3.3 The Intelligent Tutoring System (ITS) Tracking Tool .................................................. 32
    2.5.3.4 CourseVis Tracking Tool ..................................... 33
2.6 Overview on Policy-based Approach ........................................... 35
  2.6.1 Definition of Policy ................................................... 36
  2.6.2 Types of Policy .......................................................... 36
    2.6.2.1 Security Policy ................................................ 38
    2.6.2.2 Management Policy Without Considering Security .......... 39
  2.6.3 Event-Condition-Action (ECA) Model .............................. 39
2.7 Summary .............................................................................. 41

3 Computational Model ................................................................. 42
  3.1 Introduction ........................................................................ 43
  3.2 Computational Model ......................................................... 43
    3.2.1 Units of Computation .................................................. 44
      3.2.1.1 Observer ........................................................ 44
      3.2.1.2 Learning Activities .......................................... 45
      3.2.1.3 Policies .......................................................... 47
      3.2.1.4 Runtime Monitoring Mechanism .......................... 48
    3.2.2 Communication .......................................................... 49
    3.2.3 Time ........................................................................ 50
    3.2.4 Termination ................................................................. 50
  3.3 Tempura ............................................................................. 50
    3.3.1 The Language: TEMPURA ......................................... 51
    3.3.2 The Tool: AnaTempura ............................................. 55
  3.4 Summary ............................................................................. 57

4 Architecture ............................................................................. 58
  4.1 Introduction ........................................................................ 59
  4.2 Overview of The Architecture .............................................. 59
  4.3 E-learning Environment ....................................................... 61
  4.4 Tracking Layer .................................................................... 62
    4.4.1 Tracking Learning Activities ....................................... 62
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.4.1.1</td>
<td>User Information</td>
<td>63</td>
</tr>
<tr>
<td>4.4.1.2</td>
<td>Course Information</td>
<td>64</td>
</tr>
<tr>
<td>4.4.1.3</td>
<td>Assignment Information</td>
<td>65</td>
</tr>
<tr>
<td>4.4.1.4</td>
<td>Quiz Information</td>
<td>66</td>
</tr>
<tr>
<td>4.4.1.5</td>
<td>Resource Information</td>
<td>67</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Discussion Information</td>
<td>68</td>
</tr>
<tr>
<td>4.4.3</td>
<td>Observer Component</td>
<td>70</td>
</tr>
<tr>
<td>4.4.4</td>
<td>Event Repository</td>
<td>74</td>
</tr>
<tr>
<td>4.5</td>
<td>Runtime Validation Layer</td>
<td>75</td>
</tr>
<tr>
<td>4.5.1</td>
<td>Event Recogniser</td>
<td>76</td>
</tr>
<tr>
<td>4.5.2</td>
<td>Checker</td>
<td>78</td>
</tr>
<tr>
<td>4.5.3</td>
<td>Learning Activity Policy</td>
<td>81</td>
</tr>
<tr>
<td>4.5.4</td>
<td>Components’ Interactions</td>
<td>82</td>
</tr>
<tr>
<td>4.6</td>
<td>Action Layer</td>
<td>82</td>
</tr>
<tr>
<td>4.6.1</td>
<td>Action Component</td>
<td>83</td>
</tr>
<tr>
<td>4.7</td>
<td>Summary</td>
<td>85</td>
</tr>
<tr>
<td>5</td>
<td>Learning Activity Policy</td>
<td>86</td>
</tr>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>87</td>
</tr>
<tr>
<td>5.2</td>
<td>Policy Representations Based on ECA model</td>
<td>87</td>
</tr>
<tr>
<td>5.3</td>
<td>Learning Activity Policy</td>
<td>89</td>
</tr>
<tr>
<td>5.4</td>
<td>Policy Specification in Tempura</td>
<td>91</td>
</tr>
<tr>
<td>5.5</td>
<td>Summary</td>
<td>97</td>
</tr>
<tr>
<td>6</td>
<td>System Prototype</td>
<td>98</td>
</tr>
<tr>
<td>6.1</td>
<td>Introduction</td>
<td>99</td>
</tr>
<tr>
<td>6.2</td>
<td>E-learning Environment: MOODLE</td>
<td>99</td>
</tr>
<tr>
<td>6.3</td>
<td>SQL Trigger Technique</td>
<td>100</td>
</tr>
<tr>
<td>6.4</td>
<td>Prototype Structure</td>
<td>101</td>
</tr>
<tr>
<td>6.5</td>
<td>Prototype Class Diagram</td>
<td>103</td>
</tr>
<tr>
<td>6.6</td>
<td>Summary</td>
<td>105</td>
</tr>
<tr>
<td>7</td>
<td>Evaluation</td>
<td>106</td>
</tr>
<tr>
<td>7.1</td>
<td>Introduction</td>
<td>107</td>
</tr>
<tr>
<td>7.2</td>
<td>Case Study</td>
<td>107</td>
</tr>
<tr>
<td>7.2.1</td>
<td>Expressing Learning activity policies in Tempura</td>
<td>108</td>
</tr>
<tr>
<td>7.2.2</td>
<td>Runtime Validation and Supportive Feedback</td>
<td>115</td>
</tr>
<tr>
<td>7.3</td>
<td>What The Policy-based Approach Has Added</td>
<td>123</td>
</tr>
<tr>
<td>7.4</td>
<td>Discussion</td>
<td>124</td>
</tr>
<tr>
<td>7.5</td>
<td>Summary</td>
<td>125</td>
</tr>
<tr>
<td>8</td>
<td>Conclusion</td>
<td>127</td>
</tr>
<tr>
<td>8.1</td>
<td>Research Summary</td>
<td>127</td>
</tr>
</tbody>
</table>
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>CourseVis system architecture</td>
<td>33</td>
</tr>
<tr>
<td>3.1</td>
<td>Computational Model</td>
<td>44</td>
</tr>
<tr>
<td>3.2</td>
<td>Runtime Validation Process</td>
<td>55</td>
</tr>
<tr>
<td>3.3</td>
<td>Anatempura Architecture</td>
<td>57</td>
</tr>
<tr>
<td>4.1</td>
<td>System Architecture</td>
<td>61</td>
</tr>
<tr>
<td>4.2</td>
<td>Observer Component of Tracking Layer</td>
<td>70</td>
</tr>
<tr>
<td>4.3</td>
<td>Learners and Courses observation activity diagram</td>
<td>72</td>
</tr>
<tr>
<td>4.4</td>
<td>Observation activity diagram of Learning Activities</td>
<td>73</td>
</tr>
<tr>
<td>4.5</td>
<td>Snapshot of Event Repository tables</td>
<td>74</td>
</tr>
<tr>
<td>4.6</td>
<td>Event Recogniser component in Runtime Validation Layer</td>
<td>76</td>
</tr>
<tr>
<td>4.7</td>
<td>Activity diagram of Event Recognizer component</td>
<td>78</td>
</tr>
<tr>
<td>4.8</td>
<td>Checker component in Runtime Validation Layer</td>
<td>79</td>
</tr>
<tr>
<td>4.9</td>
<td>Activity diagram of Checker component’s functionalities</td>
<td>80</td>
</tr>
<tr>
<td>4.10</td>
<td>Sequence diagram of Components’ interactions</td>
<td>82</td>
</tr>
<tr>
<td>4.11</td>
<td>Action component in the Action Layer</td>
<td>84</td>
</tr>
<tr>
<td>4.12</td>
<td>Sequence diagram of Action component behaviour</td>
<td>85</td>
</tr>
<tr>
<td>5.1</td>
<td>Enforcement process of ECA-based Policy rule</td>
<td>88</td>
</tr>
<tr>
<td>6.1</td>
<td>Prototype structure</td>
<td>102</td>
</tr>
<tr>
<td>6.2</td>
<td>RTT System Prototype Class Diagram</td>
<td>104</td>
</tr>
<tr>
<td>7.1</td>
<td>Networking module in MOODLE with the learning activities available</td>
<td>109</td>
</tr>
<tr>
<td>7.2</td>
<td>Execution of Checker by AnaTempura and The enforcement of Policy-1</td>
<td>116</td>
</tr>
<tr>
<td>7.3</td>
<td>Delivering of the supportive feedback in response to the enforcement of Policy-1</td>
<td>118</td>
</tr>
<tr>
<td>7.4</td>
<td>The feedback which is resulted from the enforcement of the action in Policy-2</td>
<td>119</td>
</tr>
<tr>
<td>7.5</td>
<td>A supportive feedback displaying in response to the enforcement of Policy-3</td>
<td>122</td>
</tr>
<tr>
<td>A.1</td>
<td>Snapshot of Moodle’s database</td>
<td>149</td>
</tr>
<tr>
<td>A.2</td>
<td>Structure of Observing-Initiator.java</td>
<td>150</td>
</tr>
</tbody>
</table>
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.3</td>
<td>Events Database Diagram</td>
<td>153</td>
</tr>
<tr>
<td>A.4</td>
<td>Structure of Event-Recognizer.java</td>
<td>154</td>
</tr>
<tr>
<td>A.5</td>
<td>Structure of Event-Finder.java</td>
<td>155</td>
</tr>
<tr>
<td>A.6</td>
<td>Structure of Checker.tempura</td>
<td>156</td>
</tr>
<tr>
<td>A.7</td>
<td>Structure of Actioner.java</td>
<td>157</td>
</tr>
</tbody>
</table>
# List of Tables

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Main Tools and Features of VLEs</td>
<td>23</td>
</tr>
<tr>
<td>2.2</td>
<td>Comparison of Reviewed Tracking Tools.</td>
<td>35</td>
</tr>
<tr>
<td>3.1</td>
<td>Basic Statements.</td>
<td>52</td>
</tr>
<tr>
<td>4.1</td>
<td>User information details</td>
<td>64</td>
</tr>
<tr>
<td>4.2</td>
<td>Course Information details</td>
<td>65</td>
</tr>
<tr>
<td>4.3</td>
<td>Assignment information details</td>
<td>66</td>
</tr>
<tr>
<td>4.4</td>
<td>Quiz information details</td>
<td>67</td>
</tr>
<tr>
<td>4.5</td>
<td>Resource information details</td>
<td>68</td>
</tr>
<tr>
<td>4.6</td>
<td>Discussion information details</td>
<td>69</td>
</tr>
<tr>
<td>6.1</td>
<td>Prototype Components VS. Architecture Components.</td>
<td>102</td>
</tr>
</tbody>
</table>
## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdeLE</td>
<td>Adaptive e-LEaring</td>
</tr>
<tr>
<td>AC</td>
<td>Access Control</td>
</tr>
<tr>
<td>CBT</td>
<td>Computer-Based Training</td>
</tr>
<tr>
<td>CMS</td>
<td>Course Management System</td>
</tr>
<tr>
<td>ECA</td>
<td>Event Condition Action</td>
</tr>
<tr>
<td>ELMA</td>
<td>E-Learning Monitoring Agent</td>
</tr>
<tr>
<td>ITS</td>
<td>Intelligent Tutoring System</td>
</tr>
<tr>
<td>IV</td>
<td>Information Visualisation</td>
</tr>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force</td>
</tr>
<tr>
<td>ITL</td>
<td>Interval Temporal Logic</td>
</tr>
<tr>
<td>LMS</td>
<td>Learning Management System</td>
</tr>
<tr>
<td>MOODLE</td>
<td>Modular Object-Oriented Dynamic Learning Environment</td>
</tr>
<tr>
<td>PDA</td>
<td>Personal Digital Assistant</td>
</tr>
<tr>
<td>PRTS</td>
<td>Policy-based Runtime Tracking System</td>
</tr>
<tr>
<td>VLE</td>
<td>Virtual Learning Environment</td>
</tr>
<tr>
<td>VFTS</td>
<td>Virtual Factory Teaching System</td>
</tr>
<tr>
<td>WWW</td>
<td>World Wide Web</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Background

With the revolution of internet technology, teaching and learning using the internet has become a trend in higher educational institutes and universities all over the world (Buendia et al. 2009). Currently, a wide demand for e-learning platforms increases as the use of e-learning becomes more widespread. In order to support the growing personalised e-learning requirements, the next generation of e-learning needs to provide more flexibility and dynamism (Kovalan & Balasubramanian 2008, Graven et al. 2006, Dong & Li 2005). A Virtual learning environment (VLE) is a form of e-learning which has become widely adopted in higher educational institutions and universities (Mueller & Strohmeier 2011, TechTarget 2009, Crane et al. 2012). It offers a number of benefits, the most important of which is the flexibility of access from anywhere, at any time. Typically, a learning environment consists of three basic components: a learning management system, a set of e-contents and e-services, and a technical infrastructure (Cheung et al. 2009). VLE provides a set of integrated tools and features such as curriculum planning, self-assessment, electronic materials, tracking tools, real-time chatting, discussion forums, and so on. Learning activities are a collection of objects designed within
e-learning environments to support learners in better understanding and fulfilling the learning objectives during the learning process. Learning activities are particular activities carried out to support learners in achieving their learning objectives and qualifications. During the learning process, learners utilise and interact with different learning activities, attempting various types of assessments, collaboration, communication, and so on.

The term “Tracking” in relation to an e-learning context is the learner’s observation process of any possible interaction with learning activities. Tracking is a fundamental concept in education because without a tracking mechanism, learners may face difficulties in achieving their learning objectives. A tracking mechanism should be able to meet the learners’ needs and fulfill the stakeholder’s requirements by providing supportive and timely feedback. In traditional classroom learning, the teacher has more presence, and therefore the teacher has more control over the learners’ behaviour. All of the learners’ learning activities are done within the teacher’s view, including their attendance, adapting to the lessons, facing difficulties, their progress, etc. However, many VLEs have built-in functions to track, analyse, and report e-learners’ activities (Mazza & Dimitrova 2007). Most VLEs (e.g. Blackboard and WebCT) have similar tracking tools. These tracking tools provide reports that show the number of learners who access the course, the number of times the course is accessed, the number of attempts by each learner on an assignment, and the time spent on each attempt.

A policy-based approach is utilised to facilitate the management of a system by establishing policies to deal with events that are likely to occur. A policy has been defined as rules that are specified to regulate the behaviour of a system (Damianou et al. 2001). It contains a set of rules that reflect particular requirements. Policy can be defined based on some contexts as rules or principles that guide decisions and achieve rational outcomes (Greene 2006). In an e-learning context, it is a set of rules that ensures learning behaviour and monitors learners’ progress.
in response to learners’ interactions with different learning activities within e-learning environment.

Our research focuses on how to track the learners’ interactions with the learning activities during the learning process, how to formalise and enforce the learning activity policies using the tracked data, and how to provide a supportive action in timely manner to guide the learners and support them during the learning process.

### 1.2 Motivation

Tracking is a fundamental concept in education because without a tracking mechanism, learners may face difficulties in achieving their learning objectives. A tracking mechanism should be able to meet the learners’ needs and fulfill the stakeholder’s requirements by providing supportive and timely feedback. Tracking is an essential tool which is implemented within e-learning environments. Most current VLEs (e.g. MOODLE) are utilised with similar tracking functionalities which aim at recording statistical data for each learner, including the number of times, time, date, frequency, and IP address of each student who accesses course contents, discussion forums, course assessments, and assignments. Tracking tools also provide a summary report of individual learner performance on assignments. However, the generated statistics are module-based, which only focuses on how the contents have been accessed.

The current e-learning environment is unable to track individual learning activities where the tracked information can be used to support and guide learners. Learners do not receive supportive feedback about their learning process, and the current tracking tools do not consider the requirements of stakeholder (e.g. instructor, tutor, educational institution). A policy-based approach has been used in several Sectors, such as finance, military, healthcare, and so on, to regulate and manage the behaviour of systems, users, services, etc. (Kokolakis et al. 1998, Wu & Dube 2001, Schiefer et al. 2007, Chadha et al. 2004, Siewe et al. 2005). Therefore, a
policy-based approach can be utilised to define the way the learner behaves in a single learning activity where the policy enforcement satisfies such requirements. Tracking individual learning activities can provide useful information that can be used to enforce learning activity policies at runtime and thus provide supportive feedback. However, policy-based runtime tracking and supportive feedback within e-learning environments have not been thoroughly considered to improve the learning process. Hence, this is a motivation for this research on a policy-based runtime tracking system for e-learning environments.

1.3 Research Questions

The overall and central questions investigated by this thesis are as follows:

- *Under what strict conditions, does a policy-based tracking system provide a mechanism to track learners’ progress and the learning behaviour in the e-learning environments?*

- *In the above system, precisely what supports can be provided to learners to enhance the learning process in the e-learning environments?*

1.4 Original Contributions

The main contribution of this thesis is a novel runtime tracking system based on learning activity policies for observing learners’ behaviour and progress within an e-learning environment, so the learners can be provided with timely and relevant support. The tracking is achieved by expressing the desired learning behaviour and progress as policies, where such policies are monitored and enforced at runtime and
in an appropriate time. The major contribution is to explore and establish a novel technique for policy-based runtime tracking for e-learning environments. The following are explanations of the original contributions that will be represented by this thesis:

- **C1**: A computational model of the proposed approach is developed. The computational model will describe the nature of the units of computation and define the behaviour of such units.

- **C2**: An architecture of the policy-based runtime tracking system is designed. It has three layers including Tracking layer, Runtime Validation layer and Action layer.

- **C3**: A Learning Activity Policy is established with respect to the ECA model. A policy is formalised using Tempura where Tempura operators allow dynamic changing (i.e. at runtime) of policies.

- **C4**: A prototype of the PRTS is implemented based on MOODLE and AnaTempura. The SQL trigger technique is used to interface the proposed approach with an e-learning system.

- **C5**: A current e-learning environment (MOODLE) with a set of scenarios are used to evaluate the proposed approach. The evaluation will show the validation of the enforcement process of the learning activity policies at runtime.

### 1.5 Research Methodology

The first task when undertaking any type of research is the selection of the most suitable research method. This is important because it allows the researcher to organise the work, so the main working plan can be represented precisely to the researcher. There are several different research methods for different domains. It
is not possible to describe each one in detail, but it is appropriate to introduce some common methods of research, as follows (Goddard & Melville 2004, David J. Ketchen & Bergh 2009, McNabb 2008, Wilson 1991):

- **Quantitative Research**: This is a research method where the researcher carries out the research using techniques that make him/her independent from the research process.

- **Qualitative Research**: This is a research method where the researcher engages in activities such as conversations and observations in order to understand what people think.

- **Action Research**: This research method aims at finding a solution for an immediate problem facing society or an industrial business organisation, for example.

- **Fundamental Research**: This research method is used to improve knowledge generally, without any particular applied purpose in mind at the outset.

- **Conceptual Research**: This research method is used to develop new concepts or to reinterpret existing ones.

- **Empirical Research**: This research method usually focuses on broad empirical studies in order to verify the usefulness of known solutions on a large scale.

- **Experimental Research**: This research method always starts with some hypothesis that the research wants to test.

- **Non-experimental Research**: It is not essential to have a hypothesis using this research method.

- **Exploratory Research**: This method is used to identify new problems within a field.
Chapter 1. Introduction

The Research methods above are obviously not suitable for research projects with the focus on the development of new approaches and techniques. Therefore, it is not applicable for the majority of research projects in the domain of computer science.

For scientific research within the domains of computer science and software engineering, there is a common research method, as described below (Crnkovic 2010, Labro & Tuomela 2003):

**Constructive Research:** The constructive research method is used for the development of solutions for previously identified problems. Constructive research refers to contributions to knowledge being developed as new techniques, models, frameworks, or algorithms.

The author of this thesis, therefore, used the constructive research approach, as it is appropriate for the demands of the undertaken research. Constructive research is very often used in computer science. The research described in this thesis has been undertaken by using the constructive research method. This research method was chosen because the research project belongs to the field of computer science and the process of the constructive research considers the important tasks of development, implementation, and evaluation of the new approach. The following is a list of the steps in constructive research (Labro & Tuomela 2003). The order of the steps is adjusted for the development of the research project, as described in this thesis.

1. Select a relevant problem with enough research potential.
2. Conduct a thorough literature review.
3. Construct a solution approach.
4. Demonstrate that the developed approach works.
5. Show the research contribution of the approach.
Chapter 1. Introduction

In this thesis, the novel approach of policy-based runtime tracking for e-learning environments is proposed. Following the above steps, the proposed approach is conducted as follows:

**Step 1** is represented by Chapter 1. This chapter provides an overview on the problem of the tracking process within the e-learning environments and the motivation of the proposed approach is given. This chapter explains the research method that is used to fulfil the research represented in this thesis.

**Step 2** is represented by Chapter 2. The literature in Chapter 2 provides a thorough overview on the main topics of the research study, starting with the learning methods and e-learning environments. This chapter also investigates the tracking process in both traditional learning and e-learning. A critical review of some existing tracking approaches is provided to investigate their limitations and challenges. Finally, an overview on policies, policy types, and an ECA model is represented within this chapter.

**Step 3** is covered in Chapters 3 to 5. It focuses on developing the computational model which describes the behaviour of the system, Designing the architecture of the proposed approach, and introducing a language for specifying learning activities policies. It explores how to enforce them so that the learning activities can be tracked and supportive feedback can be provided with respect to these policies.

**Step 4** is represented by Chapter 6 and 7, where the prototype implementation of the proposed approach is covered in Chapter 6. Chapter 7 contains a case study, which is used to evaluate the proposed approach in a current learning environment (MOODLE) where a set of policies are expressed in Tempura. The evaluation shows how the objectives of the proposed approach are achieved.

**Step 5** is represented by Chapter 8 - Conclusion. The proposed approach represents a novel way to tackle the problems of the tracking and supporting processes
Chapter 1. Introduction

in an e-learning environment. The chapter shows the contributions to knowledge that are represented by this thesis.

1.6 Criteria of Success

The following criteria are given to judge the success of the research described in this thesis:

- The proposed approach should be able to provide the proposed characteristics that are lacking in the other tracking tools.
- The computational model should describe how the policy-based approach can be used in the runtime tracking mechanism.
- The proposed approach should show how the learning activity policy can be formalised in Tempura to allow for the dynamic changing of policies.
- The proposed approach should be able to provide the learners with supportive feedback at runtime during the learning process.

1.7 Thesis Outline

The following section provides a glance of the next chapters in this thesis:

- Chapter 2: Research Background
  This chapter gives an overview of the learning methods, e-learning environment, and virtual learning environment as a form of learning environment. The definition of the learning activities and the types of learning activities are described. This chapter also investigates the tracking process in both
traditional learning and e-learning. A critical review of some existing tracking approaches is provided to investigate their limitations and challenges. Finally, an overview on policies, policy types, and an ECA model is represented within this chapter.

• **Chapter 3: Computational Model**

The first part of this chapter introduces an informal computational model for Policy-based Runtime Tracking System. It describes the units of computation and represents how these units communicate and collaborate with each-other to achieve the system objectives. Communication, Timing and Termination are described in term of the execution and computation of the system components. The second part of this chapter gives an overview on the executable subset of ITL which is called Tempura and its syntax and notations. The runtime toolkit AnaTempura is also represented.

• **Chapter 4: Architecture**

This chapter provides an overview of the proposed architecture; it is divided into three layers, including Tracking Layer, Runtime Validation Layer, and Action Layer. This chapter presents the aim of the Tracking layer in the system architecture. It describes the functionalities of this layer’s components in detail, and shows how these components interact with each other. This chapter also provides a description of the information types that can be tracked from the learning environment about the learning activities. A detailed description of the Runtime Validation layer and its components is represented. The role of this layer and its components is clearly represented. A description of each component with its functionalities is also provided. Finally, the Action layer is also described with its single component (Action components). The interaction between the components in
Chapter 1. Introduction

each layer is described in this chapter using UML sequence diagrams as well.

- **Chapter 5**: Learning Activity Policy

This chapter aims at defining the policy in relation to learning activities. The first part of the chapter describes the ECA-based policy and why it is chosen. The second part represents what the learning activity policies are and how they can be represented using the ECA model. The final part describes the specification of learning activity policies in the Tempura language. Examples are represented to show how the learning activity policies are expressed in Tempura.

- **Chapter 6**: System Prototype

Chapter 6 represents the prototype implementation of the proposed approach. It starts by providing an overview of the technologies that are used. Then, the prototype components and their attributes and operations are described. This chapter also describes how these components are mapped onto the system architecture’s components. A class diagram showing the relationship between the classes of implementation is also given.

- **Chapter 7**: Evaluation

This chapter evaluates the research that is described in this thesis through a case study. It evaluates the use of the policy-based approach and the policy enforcement at runtime. It also evaluates the supportive feedback that is provided to a learner in response to the policy enforcement.

- **Chapter 8**: Conclusion

Chapter 8 summarises the research and proposes future work.
Chapter 2

Research Background

Objectives

• Provide an overview on the learning methods

• Provide an overview on the e-learning environments.

• Describe the Learning Activities in e-learning environments.

• Investigate the tracking process in some existing tracking tools.

• Give an overview on the Policies
Chapter 2. Research Background

2.1 Introduction

This chapter gives an introduction to learning methods, e-learning environments, learning activities, tracking process in e-learning and policies. It investigates a number of existing approaches related to tracking process in e-learning. Particularly, an overview on learning methods is provided in Section 2.2 which includes classroom-based learning, e-learning and blended learning. Section 2.3 describes the e-learning environments including the virtual learning environment as a form of e-learning environments. In Section 2.4, the common learning activities that are carried out within e-learning environments have been highlighted. In Section 2.5 The tracking process has been discussed in traditional learning and e-learning; however; some challenges have been addressed in a number of existing tracking systems. Finally, an overview is given on policies in Section 2.6 and the chapter are concluded by short summary in Section 2.7.

2.2 Overview on Learning Methods

The ways of delivering knowledge and instructions, and attaining them refer to which is called learning methods. There are a number of learning methods which are categorized into: Classroom-based learning, Electronic learning (E-learning) and Blended learning. Using an appropriate method depends on the instructor, learning approach, learners needs and the environment in which the knowledge is to be delivered. An overview on each learning method is carried out in the following to understand the notion and characteristics for each method.

2.2.1 Traditional Learning (Classroom-Based)

Classroom-based learning is called also a Traditional classroom learning which is a method of delivering the information by instructor to the learners face to face in a
specific place. Both learners and instructor attend teaching/discussion sessions in a physical place within a specified time. Specifically, this learning method is where the instructors focus on providing the knowledge and learners concentrate on acquiring it (teacher-centred). The role of the instructor within classroom-based setting is to deliver information to learners, assist learners and observe learners’ behaviour and progress. However, without the teachers’ potential to distribute the knowledge, learners can hardly acquire that knowledge and therefore, the learning process can hardly take place. The most responsibilities of learning process lies on the instructors within traditional learning method. The knowledge delivery in classroom-based learning is limited due to the limitations of the resources available within the classroom or the educational institution. Most of the assessments made within this learning setting relies on what was delivered by the instructor which in turn limits the knowledge acquisition of learners (Li et al. 2005).

This learning method has to take place within a specific area, however, there is always a restricted geographical distribution. Learning process is always taken place at particular time, place and particularly with a known instructor. It was suggested by Chickering et al. (1987) that learners should not limit themselves to what they listen in the class but also should read, discuss, or be engaged in finding solutions. There is a need for alternative learning methods that offer learners more freedom to construct knowledge at their convenience. To meet the learners needs with responsibilities and more features that are hard to be available in physical classroom, another learning method should be developed to tackle these problems and provide more features. The paradigm shift from having the teacher-centred to more learner-centred learning . Therefore, the developments in technology resulted to a learning setting which is called e-learning. The next section gives an overview on this method which is adapted by many of educational institution around the world.
2.2.2 E-learning

A lot of researches (Clark & Mayer 2011, London 2011, Tomic et al. 2011) describe the new path of learning which is called electronic learning (E-learning) using different technologies. The concepts of education and business have been used to describe the term of e-learning. The terms “Electronic” and “Learning” are conceptualized by different researchers to define this method of learning. Many researchers (Ma et al. 2010, Babic 2011, Hasibuan & Santoso 2005) agree to that it is the learning that is curried out using technology to enhance the learning process. E-learning is the method of delivering the knowledge in a synchronous (i.e. Real-time) or asynchronous (i.e. self-paced) format using electronic tools such as World-Wide-Web (WWW), Intranet, DVD, CD-ROM, TV, etc.. The definition of the term of e-learning can be generalized as the way of delivering the knowledge and training using any electronic media (Hayakawa et al. 2012, Zimmermann 2011).

Furthermore, the e-learning method can be described as the use of computer and network technologies to create, deliver and facilitate learning, so learners can access and learn at any time and from any-place. E-learning was defined by Meyen et al. (2002) as the delivering and acquisition of the knowledge which is dispensed and facilitated essentially by electronic tools and devices. It was argued that despite the e-learning environments rely on computers and networks technologies, different channels are also utilized these days in e-learning systems such as wireless, smart phones, mobile computers (PDAs), etc.. (Ivan Ganchev 2008). E-learning provides facilities to learners allowing them to learn anywhere, at any-time and at their convenience. This leads the learners to have more flexibility in term of knowledge acquisition. The e-learning environments should be flexible in assisting all learners during the learning process. E-learning systems are usually designed with more facilities that support learning process and make an efficient way to learn.
Chapter 2. Research Background

Nowadays, e-learning systems have been adopted by many educational institutions, companies and industry sectors around the world for education and training purposes (Buendia et al. 2009). Using the e-learning method can enhance the learning process and learner-centred learning. The e-learning setting puts the learner in the centre of learning process, hence, giving more opportunities for knowledge acquisition and collaboration (Zhang et al. 2007).

2.2.2.1 Types of E-learning

Different categories of e-learning method have been described by Siemens (2004) as used by the community of e-learning. There are four common types of e-learning (Consulting 2012).

- **E-learning Conventional Type**: This type involves a normal group training where the knowledge is delivered by using electronic equipments such as TV and Radio. This type can also be carried out in training classroom using a computer to deliver power point slides.

- **Computer-based Training(CBT)**: This type of e-learning is called also off-line learning. It occurs using a computer and DVDs. The learners use their computer to fulfil special training programs using; for instance; tutorial DVD. The common courses that are provided by CBT is learning foreign languages using special software and CDs.

- **Asynchronous E-learning**: In this type of e-learning, the Internet is utilised to deliver an on-line learning process. This type has become popular recently due to the improvements in the technologies. Asynchronous e-learning is very flexible because it can be carried out at any-time and from anywhere by both the instructors and learners. In this type of e-learning different resources can be supplied to the learners such as e-books, video lectures, etc., that contain the learning contents. Learners can communicate with each other or instructor through emails, messaging board and so on.
Chapter 2. Research Background

- **Synchronous E-learning**: This kind of e-learning is called also real-time learning due to its facilities that allow the learners and instructors interacting with each other and learning contents at the same time. Chat-room and video conferencing are used to carry out real-time discussion sessions. Learners can view and communicate with other learners through the e-learning environments. Instructor has control over the learning process and can observe the learners’ progress. The e-learning environments of this type of e-learning let learners utilise many types of learning activities including assessments, content engagement, collaborative discussions and so on, which enhance the knowledge acquisition and learning process.

### 2.2.3 Blended Learning

According to (Valia 2002), blended learning method is the way of learning that combines a mixture of different learning styles including face-to-face classroom learning, real-time learning activities and off-line learning activities. This learning method mixes the class-based learning with e-learning. This method was developed due to existing problems found in both e-learning and traditional learning settings. The blended learning is described as a hybrid where the learning process is carried out in both classroom and web (Collis & Moonen 2001).

Blended learning was defined by Garrison & Vaughan (2007) as a way of delivering knowledge at a distance that uses technologies such as (learning systems, internet, etc..) combined with face-to-face traditional learning. This kind of method contains different tools to provide flexible rich learning environments. These tools may include on-line contents, textbooks, multimedia, digital libraries, chat rooms, discussion boards, smart classes, etc..

Blended learning is a result of a paradigm shift that occurred in many universities and other educational institutions to produce more learner-centred environments (Barr & Tagg 1995, Cheung et al. 2010). This allows learners to acquire the
knowledge, and teachers to become instructional designers and moderators of the learning process. Blended learning method offers a chance for learners to control what they missed in the traditional classroom. Some instructors found that this method is better in delivering specific content. Some learning contents is appropriate with an on-line learning style and other need to be delivered in classroom.

Blended learning can be carried out in various ways depending on the learning needs. Combining different learning styles may include blending on-line instruction with classroom instruction, blending on-line instruction with instructor for private training, etc. Hence, for any blended learning to take place, different learning styles are combined to increase the effectiveness in the learning process and the knowledge acquisition. Blending the classroom-based learning with on-line learning can create more individualised way of learning and, therefore, lead to more flexibility in: a) the knowledge delivering by instructors and, b) accessing by the learners. The blended learning method is not a replacement of either the traditional classroom learning or e-learning, but it is a mixture that offers new and more effective experience of learning process (So & Brush 2008).

2.3 E-learning Environments

With the revolution of the internet use and its services, and with evolving of e-learning systems with World Wide Web, teaching and learning using the internet has become a trend in modern higher educational institutes and universities all over the world these days (Buendia et al. 2009). Currently, a wide demand for e-learning platforms is increasing as the use of e-learning becomes more widespread. In order to support today’s growingly personalised e-learning requirements, the next generation of e-learning needs to provide more flexibility and dynamism (Kovalan & Balasubramanian 2008, Graven et al. 2006, Dong & Li 2005b).
E-learning systems are software applications which are “multi-disciplinary systems requiring a group effort, where educators, administrators, and users from a variety of other areas of expertise (e.g. psychologists, sociologists, software and electronic engineers) come together in order to support a community of learners” (Al-Ajlan 2008). These application are designed to assist instructors in order to meet their pedagogical goals of delivering learning contents to a community of learners (Al-Ajlan & Zedan 2008). In the last decade, virtual learning environment (VLE) as a form of e-learning has become widely adopted in higher educational institutions and universities. It offers numbers of benefits with the most important one the flexibility of access, from anywhere, at any time. Typically, a learning environment consists of three basic components: learning management systems, set of e-contents and e-services and technical infrastructure (Cheung et al. 2009).

2.3.1 Virtual Learning Environment (VLE)

In recent years, the use of e-learning systems has shown sufficient growth in higher educational institutes and universities in visualising learning and teaching through effective environments benefiting from the growth of the internet and communication technologies (Toth et al. 2006). Virtual learning environment is a form of e-learning that provides set of integrated tools and features such as curriculum planning, self-assessment, delivering electronic materials, tracking tools, real-time chat, discussion forums and so on (Dong & Li 2005a, Chou & Liu 2005).

In addition, there are a rich set of benefits offered by VLEs in which the most important one is flexible anywhere, any time access. More benefits are offered by VLEs such as encouragement of collaboration and communication, active engagement and better integration of computers, communications techniques, tools and a community of learners (Georgouli 2011). Virtual learning environment is a form of e-learning that provides set of integrated tools designed to support the learning process for both instructors and learners and to enhance learners’ learning
Chapter 2. Research Background

experience (Georgouli 2011, Vilarino & Garcia 2011). The principal components of a VLE package include curriculum mapping, learner tracking, on-line support for both teacher and learner, electronic communication (e-mail, threaded discussions, chat, Web publishing), and Internet links to outside curriculum resources (Piovesan et al. 2012).

According to Berry (2005) “A VLE is an electronic system that can provide on-line interactions of various kinds that can take place between learners and tutors, including online learning”.

VLE software is currently being used across most UK institutions to support a variety of different types of learning: for example, collaborative and co-operative, blended and distance learning. In other countries, such as the United States, Australia and some European nations, VLEs are more commonly referred to as ‘Course Management Systems (CMS)’ or ‘Learning Management Systems’ (LMS). Many commercial VLEs (e.g. Blackboard, WebCT etc.) consist of a variety of different tools bundled together into a package. There are also a number of open source software packages available (e.g. Moodle and Bodington) (TechTarget 2009).

VLEs have also been defined by Britain & Liber (1999) as “learning management software systems that synthesise the functionality of computer-mediated communications software (e-mail, bulletin boards, newsgroups etc.) and on-line methods of delivering course materials (e.g. the WWW)”. To date, several different packages have appeared from both leading commercial vendors and university-based projects (Britain & Liber 1999). With use of technology in teaching and learning, VLEs is beneficial to education. VLEs can help higher education with increasing number of learners. Flexibility of access any where and at any time is a major benefit of virtual learning environment. VLEs allow participants to stay at their convenient time and location. The way in which VLEs are designed and delivered
can add value and increase effectiveness. There are actual and quantifiable benefits gained from the use of VLEs and that makes these technologies so popular and embedded in many institutions nowadays (Piovesan et al. 2012, Jaligama & Liarokapis 2011). The following are some common advantages of using VLEs as reported in (Al-Ajlan & Zedan 2008, Jaligama & Liarokapis 2011):

- **Advantages**

1. A VLE offers the instructors tools to encourage collaboration and communication. For example, a VLE can provide a virtual space where learners, staff and other learning support specialists can discuss, interact, share learning, ideas and materials.

2. Tutors can benefit from the ‘administrative tools’ within a VLE. Many VLEs provide information to staff about how often and when learners have accessed a VLE through the tracking tool. They may also provide information about when and what they have read in the online discussion area.

3. It is often difficult to find time or a way to ensure that learners actively participate in face-to-face sessions. Through online discussions it is possible to help learners engage more actively with a course and with the learning process at a time and place that is convenient for both tutor and learner.

4. The result of this collaboration and communication may be to develop a unique space which the learners’ cohort builds into its own identity and community: a community of learners.

5. Through careful course design, tutors can support the communication and collaboration in a VLE with specific signposting and access to a vast array of up-to-date, multimedia, interactive online materials for learners. This can be material that is developed by the tutor, for example, lecture notes, diagrams and images. It could include links to web resources, the institution’s online
library resources, web resources developed by publishers for core texts, online articles, graphics or searchable online databases.

- The main features and Tools of VLEs

Virtual learning environment as a single software product is combining the following as reported in (Dunn 2003):

- communication tools (e-mail, bulletin board, chat room, etc.).
- collaboration tools (online forums, files-sharing, diaries, etc.).
- content creation tools.
- assessment tools and tracking tools.
- integration with institutional management information systems.
- controlled access to curriculum resources.

VLE tools are criteria-based, and they enable developers to evaluate and select the most appropriate VLE products. No single product can possibly meet all these criteria, and the most appropriate within a specific context may not be perfect for interface, technical, functional, or cost reasons. Table 2.1 describes tools and features which should be considered when VLE is to facilitate a complete learning and teaching experience (Al-Ajlan 2008, Cheung et al. 2009, Dunn 2003, Britain & Liber 1999, Cheng & Yen 1998, OLeary & Ramsden 2002, Dougiamas 2002, JISC 2011).
Chapter 2. Research Background

2.4 Learning Activities in E-learning

Learning activities are a collection of objects designed to support learners having better understanding and fulfil the learning objectives during the learning process. Learning activities are particular activities designed to support learners achieve their learning objectives and qualifications. During learning process, learners utilise and interact with different learning activities attempting various types of assessments, collaboration, communication and so on. Learners interactions with these learning activities indicate what knowledge may be acquired. In this thesis, we focus on the common learning activities that are carried out in most of e-learning systems which include assessment, communication and collaboration and content-interaction.
2.4.1 Assessment

Assessment is the important way in education to measure the achievement in the knowledge acquisition process. An assessment form describes the learners’ performance over the time in the learning process (Broadfoot et al. 2001). The assessment was described by Elwood & Klenowski (2002) as tool for learning where the purpose is to let learners understand their learning outcomes and objectives they aim for, through effective feedback. The purpose of the assessment is not only for grading the learners but also to guide them during the learning process. The assessment tools are used in the e-learning environments to facilitate the learners effectively during the learning process.

Assessment is the means used to assess learners understanding, skills abilities and attitudes by carrying out quizzes, assignments, exams and surveys. The most important types of assessment that are carried out during learning process as described by Shepherd & Godwin (2010):

**Formative Assessment:** This kind of assessment is described as a set of quizzes or assignments that are taken within specific topic before the final examination of the module. The aim of this type of assessment is to provide learners with practice in retrieval from memory in relation to acquired knowledge and also to provide supportive feedback. Formative assessment plays an important role in identifying and ensuring learners’ progress during the learning process (Torrance & Pryor 1998). The results of this kind of assessment can be used to provide feedback about the learners progress.

**Summative Assessment:** This kind of assessment is called also a quantitative assessment which aims usually at giving a definitive mark and/or grade as a judgements of the overall achievement. The learner is considered to be passed when the judgement verifies that learners has met the established objectives. This type of assessments is carried out in form of exam or assignment report to determine the overall understanding of the module. In other words, this type of
assessment is used to making a final decision on the learners’ achievements and the knowledge acquisition process. Stakeholders (i.e. Instructor, learner, course designer, etc.) may use the summative assessment as evidence for accountability on deciding whether or not the learning was effective (Hao & Fanghua 2009).

However, Torrance & Pryor (1998) pointed out that it is not only important to identify what learners have achieved but also what is not achieved and what can be achieved in future. Through assessment, the learner progress and achievement can be observed to know the objectives accomplished and not accomplished. Providing effective feedback on the output of the assessment is important as it guide learner to the way forward during the learning process. This is because the main goal of feedback is to support learner and improve their progress, hence feedback should be provided on timely manner when it is more relevant (Gipps 1994).

There are a number of methods and types of assessment that have been developed within the e-learning. The use of quizzes in different styles such as multiple choice, True/False, options and match questions, are the most popular in many e-learning environments. The use of these kind of questions has greatly increased the efficiency in the assessment within e-learning. Therefore, learners can be assessed in real-time and effective feedback can be offered for more improvements.

2.4.2 Communication and Collaboration

Communication and collaboration are learning activities that are utilised during the learning process and involves one or more learners who communicate with each other through different tools such as chat-rooms, discussion boards, video conferencing, emails, etc. This type of learning activities is offered asynchronously and synchronously and involves who communicate through the offered tool with the people involved in the collaboration process (Dillenbourg et al. 1996). Roschelle & Teasley (1994) described the collaboration process as the involvement of shared engagement of learners in a coordinated effort to solve a particular problem. The
collaborative communication between learners lead learners to share the knowledge and their understanding in particular topic. Therefore, the collaboration learning activity offers an opportunity to share understanding of other participants’ view on a particular topic during the discussion. It has been pointed out by Prawatt & Floden (1994) that sharing knowledge by a group of learners in finding a solution of specific problem is much easier than solving the problem individually. When number of people share ideas about the problem, they find out the proper solution which comes through collaborative search. Collaboration can be performed between learner-learner, learner-instructor.

In learner-instructor collaboration, the instructor carries out and tracks the collaborative discussion among the learners on which creative thinking and ideas are being presented to find out the solution of the problem. This kind of collaboration enables the learners to achieve an understanding of the problem in social manner. So, a shared social experience and social argument of the ideas are involved during the collaborative discussion.

Several researchers have shown that allowing collaboration in learning leads to effective learning process (Peterson & Janicki 1979, Light 1993, Webb 1982, Jonassen et al. 1995, Berge & Collins 1995, Kanuka 2011). Collaborative discussion within a group of learners in finding out the solution of particular problem saves learners’ time. Collaboration that is offered by e-learning environment supports the shy learners during learning process because the collaboration occurs without physical presence. This kind of learners may be encouraged to engage the discussion and show the ideas without being noticed. This increases the performance and confidence of learners during learning process.

Collaboration and communication is essential in any effective learning environment because it offers learners a chance to discuss, share ideas, and understand the problem to be solved. An effective discussion should be carried out based on a clear purpose to the learners. In e-learning environments, collaboration and
communication facilities play a great role in enhancing the learning knowledge acquisition during learning process.

### 2.4.3 Contents-Interaction

This is another common type of learning activity that involves learners interacting with different learning contents to attain new knowledge and skills. Learners interacts with learning contents using their own preference. Thus, it is important that all learner should be provided with suitable learning contents that satisfy their needs. Due to the growth of technology developments, e-learning environments provide learning contents that suit all learners (Zhang et al. 2004). Within the leaning objects, instruction can be created, so, different learning contents can be suitable for different learners (Zhu et al. 2010).

### 2.5 Tracking Process

The term ”Tracking” in relation to e-learning context is the observation process of any possible interaction with learning activities that learners make during the learning process in order to support and assist learners effectively to achieve their learning objectives.

#### 2.5.1 Tracking in Traditional Learning

Despite in traditional learning the main centre of knowledge acquisition is instructors, they carry out particular tasks to monitor and support learners during the learning process. In traditional classes, the teacher has more presence (France et al. 2006), therefore the teacher has more control over the behaviour of the learners. All of the learner activities are always within the teacher’s view, including attendance, how learners adapt to the lessons, what difficulties the learner face,
learner progress, etc. A teacher in a conventional classroom setting is able to notice the body language of the learners. Therefore, the teacher can immediately respond to non-verbal cues, such as a reduction in learner interest or an unenthusiastic performance (e.g. a learner spends more time than normal performing a task, a learner does nothing, or a learner does not collaborate with his or her peers when working in a group. Additionally, teachers in a traditional classroom setting are conscious of class unity, such as how well the learners understand the lessons, how many learners did the coursework, how many resources the learners utilised, who succeeded at completing a certain task, and so on. Teachers in a traditional setting are normally required to ensure that their learners meet all requirements of the course they are taking by being proactive in learner development in a timely manner. Because traditional teachers instruct their learners as a group, their learning sessions are more interactive: learners can help each other, the teacher can conduct organised discussion groups, and the teacher and the learners can come to a mutual understanding or a group consensus over certain activities (Alghamdi et al. 2011).

2.5.2 Tracking in E-learning

The major aim of e-learning environments is to allow learners access to the course at the times and locations which are convenient for them, which, in turn, encourages them to pass the course. Learners participating in e-learning have the freedom to study and participate in all activities. The drawback of this flexibility and freedom is that instructors cannot personally interact with the learners as a group, so the performance and behaviour of the learners is not as easily controlled, and the absence of class unity may discourage an individual learner’s sense of accountability to the course policies. However, most VLEs have a built-in functionality to track, analyse and report e-learners’ activities (Mazza & Dimitrova 2007). Most commercial VLEs (e.g. Blackboard and WebCT) and free open source VLEs (e.g. MOODLE) have similar tracking tools, which provide reports
that include the number of activities performed by learners and the types of actions they have taken. The following list presents some features of the learner tracking tool in MOODLE as reported by Shu Chien (2009):

- It provides reports that show the number of learners who access the course, the number of times the course is accessed, the date of each access, and the IP address of each learner who accessed course contents, discussion forums, course assessments, and assignments.

- It provides a report that shows the number of attempts by each learner on an assignment and the time spent on each attempt. Instructors can keep private notes about each learner in a secure area.

- Learner tracking also provides a summary report of each learner’s performance on all assignments.

- The instructor can set a flag on particular course components to track the frequency that learners access those components.

- It allows the instructor to notice the learners who are currently logged in to the course.

The features above show statistical data from large logs of learners’ activities, but the problem is that this comprehensive data that is presented to the instructors is the burden of interpretation. However, such tracking tool does not provide any form of feedback which may support the learners. Instructors also must spend an enormous amount of time studying the statistics in order to translate them into behaviour patterns before they can provide appropriate feedback to the learners.
2.5.3 Review on Existing Tracking Tools

There are many works and research that focus on instruments and methods proposed for assessing learners’ learning style. This led to some solutions to be proposed by researchers to support assessments and learning process and producing tracking and monitoring systems and tools for e-learning environments to track and observe learners activities and the use of contents. In this section, we investigate some tracking systems and tools which were proposed and developed to be integrated to an e-learning environment (e.g. MOODLE) or to be as stand-alone systems.

2.5.3.1 E-Learning Monitoring Agent (ELMA) Tracking Tool

ELMA is a tracking tool which was developed in Indiana Polis, USA and integrated within ANGLE e-learning environment (Schneider 2005). ELMA tracking tool contains software agents used to carry out monitoring functionalities of different learning activities utilised by the learners. The main concern in this section is the tracking process carried out within ANGLE. Within ELMA, there are different software agents that perform different functionalities. The WhoDunIt agent aims at monitoring learning activities performed by learners. It monitors learners’ interaction with learning activities. Specifically, WhoDunIt agent monitors who perform learning activity and when, accesses to learning materials, attempts of assignments and so on. The course tutor guides the WhoDunIt agent to track and analyse the interaction information of learning activities. The tutor selects specific tools designed within the e-learning environment to guide this software agent. The tracking information may include learners’ login details, completed tasks, assignments results and discussion activity. This information is stored under the learner profile. Another agent implemented within ELMA is responsible for analysing the stored information and representing the reports about performance measures and statistical analysis for each learner.
Chapter 2. Research Background

The feedback about learners’ progress is represented in the form of visual charts. These charts describe grades and access of learners as compared to the class average. A supportive feedback is not offered in response to the learners’ interactions with learning activities. The learners and educational requirements in relation to the learning activities are not fully considered in ELMA. Therefore, these limitations need to be addressed in order to improve ELMA tracking tool. The tracking tool should be able to offer supportive feedback in response to learners behaviour on utilising a learning activity. The time should be also the main concern when developing tracking tool, so assistance should be provided to the learners in appropriate time.

2.5.3.2 AdeLE Tracking Tool

Adaptive e-Learning (AdeLE) is essentially an Eye-Tracking tool used as a tracking tool within Adaptive Knowledge Transfer System (Guetl et al. 2004). AdeLE combines real-time eye tracking with content tracking to make a learning profile. This tool basically monitors the motions of a learner during the learning process. AdeLE undertakes activities that capture the learner’s behaviour dynamically based on real-time eye-tracking (Pivec et al. 2006). A special camera is placed on learners’ computers that is used for learning to track all learner’s motions during learning process. The tracing data is stored under the learner profile for further processing. So the main aim of this tool is to observe the learner’s motion, so for instance if the learner appears to be drowsy then feedback will be offered indicating that he or she needs to pay more attention. The technique in this tracking tool concerns with observing learner keyboard clicks, mouse cursor movements, body movements and the time the learner spends on a specific page.

The AdeLE tracking tool uses analysis and evaluation of real-time eye and content tracking in order to support and encourage learners during learning process within e-learning environment. A learner profile is created in this tool using tracking data containing personal characteristics, momentary states and content consumption.
The tool uses learners’ learning behaviour to adapt them with suitable learning content. There are some challenges which should be addressed within this tracking tool. The learning mood of learners may change over the time, hence adapting to misleading observed movements data is not initially considered when offering learning content. As this tool requires a camera to be placed on the learners’ computers, hence this leads to some restrictions, so the learner may not be able to access the learning environment from another computers. Moreover, the privacy issue is not considered as some learners may oppose to this kind of tracking technique which barges on their privacy.

2.5.3.3 The Intelligent Tutoring System (ITS) Tracking Tool

The Intelligent Tutoring System (ITS) is a tracking tool which employs agents to monitor learners during the learning process. It was developed in California, USA using artificial intelligence technology (Graesser et al. 2005, Beal 2004). In ITS tracking tool, the animated pedagogical agents are employed to track the learners during their interaction. The main goal of ITS tool is to monitor learners’ activities and detect difficulties they may face during the learning process. The agent’s role is to track interface actions, focus of eye gaze and the data used in initiating other future tasks. A Bayesian network was used to reason about the learners’ focus of attention and forming probabilities of future actions.

Virtual Factory Teaching System (VFTS) user interface is used by ITS tool and designed specially to monitor the keyboard entries and mouse clicks performed by the learners. A camera is positioned to focus on the learner’s face and capture the motions. The difficulties the learners may face during performing specific task are detected by the agents, however, the agents offer assistance to overcome difficulties. The eye gaze is a point of focus within the ITS tracking tool because the eye gaze describes the state of learner, so help can be offered. This is similar to the eye-tracking but the software agents are employed in decision making on the type of help to be provided.
Chapter 2. Research Background

The assistance provided by ITS tracking tool is in form of suggestions about the appropriate learning contents. The feedback offered by this tool is not related to the accomplishment of learning objectives. The tracked data about the learner movements may be inaccurate. The learners and educational requirements are not considered when offering feedback, however this is a challenge that needs to be considered for more effective feedback and learning improvements.

2.5.3.4 CourseVis Tracking Tool

According to (Mazza & Dimitrova 2007, 2005), CourseVis is an approach proposed to graphically represent learner tracking data (web logs) generated by a VLE. A technique from information visualisation (IV) was employed to graphically provide complex, multidimensional learner tracking data. CourseVis (see Figure 2.1) has been designed as a tool that extends an existing VLE by adding new functionality in terms of graphical representations produced with OpenDX which is an open source and generic visualisation tool.

![CourseVis system architecture](image)

**Figure 2.1:** CourseVis system architecture (Mazza & Dimitrova 2007).

CourseVis is a tracking tool which has been developed as a generic tool for visualising learner tracking data. It is required that the data from a VLE is converted
into an XML format, which is then imported into the CourseVis graphical render to produce various representations of the data. The main aim of CourseVis is to help instructors become aware of the social, behavioural and cognitive aspects of e-learners (Mazza & Dimitrova 2007, 2005, Atterer et al. 2006, Mazza & Dimitrova 2004). However, as shown in the CourseVis system architecture (Figure 2.1) and its functionality, this system makes learner tracking data, which are generated by VLE, to be more understandable by instructors, by transforms them into graphical representations.

However, the factor of time-consuming is not considered within this tool. The instructor may spend enormous amount of time to follow the graphical charts and representations provided by this tracking tool to know the learner progress and level of achievement. There is a challenge that needs to be addressed within CourseVis which the inability to provide supportive feedback in response to learners interactions with leaning activities.

Different functionalities are reviewed in the tracking systems above which are employed for monitoring the e-learning environments. Table 2.2 illustrates a comparison of the main functionalities addressed in the reviewed tracking tools above. The purpose of this comparison is to show how efficient the tracking systems above are in relation supporting learners during the learning process in the e-learning environments.

A number of challenges were addressed in the reviewed tracking tools above, so they need to be considered when designing a tracking system. The addressed challenges can be summarised as follows:

- The need to monitor learning activity interactions made by learners which initially depends on the requirements of stakeholder (e.g. tutor, course designer or educational institution).
Table 2.2: Comparison of Reviewed Tracking Tools.

- The need to use the policy-based technique where such requirements can be represented by a set of policies.

- Enforcing such policies at runtime to offer learners with supportive feedback in a timely manner for enhancing the learning process and the learners’ progress.

2.6 Overview on Policy-based Approach

Policy is a set of rules that reflect particular requirements that may change over time. Policy-based approach has been used in several sectors such as finance, military, healthcare and so on, to regulate and manage the behaviour of systems, users, services, etc., (Kokolakis et al. 1998, Wu & Dube 2001, Schiefer et al. 2007, Chadha et al. 2004, Siewe et al. 2005). Current tracking tools as mentioned above do not consider the requirements of stakeholder (e.g. Instructor, tutor, educational institution). Therefore, policy-based approach can be employed to define the way the learner behave on single learning activity where the policy enforcement satisfies such requirements. This section gives an overview on policies, specifically the definition of a policy, the classification of policies and the types
of policies. Event-Condition-Action (ECA) model is described also as a type of policy.

2.6.1 Definition of Policy

A policy was defined as a set of rules that is specified to regulate the behaviour of a system (Damianou et al. 2001). It contains a set of rules that reflect particular requirements. Policy can be defined based on some context as rules or principles which guide decisions and achieve rational outcome (Greene 2006). According to the organisation of Internet Engineering Task Force (IETF) (IETF 2012), the policy is defined as a definite aim, path or technique of action to guide and determine present and future tasks. Saperia (2002) has defined the policy as

“A predetermined action pattern that is repeated by an entity whenever certain system conditions appear.”

As mentioned above, the term ‘Policy’ has been defined variously and the reasons behind are that the policy may be defined depending on the applications’ nature in which policy is being enforced and the availability of different policy tools (Chadha & Kant 2007). There are three common types of policies which are security, access control and management policy. These categories of policy may be deployed in several sectors such as commercial, finance, learning, military, healthcare and so on.

2.6.2 Types of Policy

The policies may be classified based on their specification within a system, or the properties which the policy should satisfy. According to Chadha & Kant (2007), policies can be classified as follows:

- Policy as Rule Dictating Behaviour
This type is the most common used kind of policy. The policy rules of this type are procedural in their nature (i.e. the policy indicates what specified action should be executed under particular circumstances). The policy rules in this type dictate the behaviour of the system. For instance:

- Upon system infection, if the virus is removed, then do a back up.
- Everyday at 2 a.m do system back up.

In first example, the event that is declared is when system is infected, a condition should be checked if the virus is removed, and if True, then execute the specified action (back up). In second example, the rule is specified with event of (when Time = 2 am every day) then the action should be executed (system’ back up) with no specific condition.

- Policy as Rules Granting or Denying Permissions

The rules of this kind of policy are declarative. They are used to specify grant or denial permission of action to be performed on specific targets by specified objects. For instance:

- Allow Bob to Write on (File X).

The specified action in this example is (Write), the object is (Bob) and the target is (File X).

- Policy as Constraints or Parameters

These types of policy contain rules that are specified as constraints or parameters that regulate the system. The rule’s constraints are specified in a declarative fashion. Constraints provide declaration statement regarding a particular relationship that must holds TRUE in a system at all times. These constraints are also called a policy because they dictate behaviour of the system. Such policies are used
to provide high-level targets and objectives to be reached. The following is an example of constraint policy:

- *Network traffic should be monitored every 60 seconds.*

Policies can be categorised in term of their functionality. The following describes two common types of policy: Security Policy and Management Policy.

### 2.6.2.1 Security Policy

Security policy is the statement that divides the states of the system into a set of authorised (or secure state) and unauthorised (or unsecured states) (Bishop 2002). According to Dong oh et al. (2006), security policy defines the condition of an authenticated subject (user) is allowed or denied to perform a particular action on a specified object (target).

Access Control (AC) policy is a popular security policy used to control the access to a system. There are several types of access control policy such as:

- **Authorisation policy**: defines what activities a subject can undertake on a set of objects. This type represent the essence of access control policies. The authorisation in this type can be positive or negative. The positive authorisation define the allowance of performing the action on specific object by the subject, and the negative authorisation is the denial.

- **Delegation policy**: defines the condition under which a subject can delegate a specific access right to another subject.

- **Obligation policy**: defines the conditions or events (sometimes also called triggers) under which a subject has to perform a specific action on an object.
2.6.2.2 Management Policy Without Considering Security

A management policy is used to specify dynamic adaptable management strategies, so they can be easily modified to change the management approach without affecting the management system. According to Damianou et al. (2002), most management policies are specified in form of Condition-Action rules, so they can be easily modified because they simply contain conditions and actions. IETF (2012) proposed a policy model as management policies, so the rule syntax is as follows:

\[
\text{If } \langle \text{condition(s)} \rangle \text{ then } \langle \text{action(s)} \rangle
\]

The part of condition in the rule can be described in a simple disjunctive normal fashion or compound conjunctive expression. The rule's action part can be one action or set of actions that must be executed when the conditions are TRUE. The rule form of this type of policies is similar to one in the obligation policy, however, there is no obvious event specification to trigger the execution of the specified action. Instead, it is supposed that an obvious event will trigger the policy rule such as a user request.

2.6.3 Event-Condition-Action (ECA) Model

ECA model is used by many researchers as a kind of policy that simply contains an event, condition and action. Generally, it is a simple pattern used to specify particular requirements. ECA policy is represented by a set of rules that execute a specified action in response to the occurrence of events and a condition holds TRUE. Such simple representation leads to a reactive functionality of a system to be specified and maintained within a single rule base. Hence, this enhance the modularity and maintainability of the system (Papamarkos et al. 2003). The General syntax of an ECA rule is as follows:
As shown above, the ECA rule comprises of the following parts (Poulavassilis et al. 2006):

- **Event**: provides the means of policy trigger. Event occurs asynchronously in the relevant domain (i.e. environment) of a system.

- **Condition**: is a part of ECA rule used to check whether a specific rule’s condition expression is set to TRUE, so the action specified in the rule is to be executed.

- **Action**: is the main part of the rule as an operation that specified to be taken when specific event occurred and after a rule condition is set to TRUE. An action can be used also to fire another ECA policy.

However, the event and condition are optional parts in ECA where the action part is compulsory in an ECA policy.

ECA model can be used to represent almost any policy rule especially management policies. As described previously, the management policies are usually represented in form of IF-Then rules, thus they are specified by means of an ECA pattern. For instance, assuming that a policy says that ‘On system infection, if the virus is removed, do back up’, using ECA model to represent this policy can be represented as follows:

\[
\text{On system infection (Event), If Virus is removed (Condition), Do back up (Action).}
\]
2.7 Summary

In this chapter we firstly gave an overview on learning methods. The author then reviewed the e-learning environments as the VLE is our concern in this thesis. Three types of learning activities are highlighted in Section 2.4 where the information that can be tracked from these learning activities is described. The tracked information can be used to provide learners with the needed support when performing the learning activities. A critical review also is provided on the tracking process. The tracking process in the traditional classroom has shown that it is efficient because the learner can be monitored continuously by the teacher and the support are given immediately, so the problems that the learner may face can be solved in appropriate time. Several limitations and challenges are addressed related to the tracking process in both the built-in tracking tools in most VLEs and some existing tracking tools that were developed. Using a policy-based approach in the tracking process provides an effective tracking mechanism which will be described in the next chapters. Enforcing a policy at runtime in response to learners’ interactions with a learning activity leads to provide learners with supportive feedback in timely manner. For this purpose, an overview is provided on the policies in Section 2.6. The next chapter describes our computational model of Policy-based Runtime Tracking on which the rest of this thesis is based.
Chapter 3

Computational Model

Objectives

• Describe the computational model of PRTS.

• Provide the analysis of learning activities.

• Give an overview on Tempura and its tool AnaTempura.
Chapter 3. Computational Model

3.1 Introduction

This chapter aims at providing an informal description of our computational model for Policy-based Runtime Tracking System. The computational model describes the nature of components of software systems, behaviour of such components and provides the basis to reason about properties of the individual system components’ behaviour and their interaction. The next section describes a model of computation for PRTS including the units of computation and a description of the notion of communication, timing and termination of such units. A short analysis is shown about the learning activities under the computational model section. Section 3.3 provides an overview on Tempura and the runtime validation tool-kit AnaTempura where the Tempura will be used to formalise our learning activities policies and its tool AnaTempura will be used as runtime validation engine which interpret and enforce the policies at runtime.

3.2 Computational Model

Computational model describes the nature of components of software systems, behaviour of such components and provides the basis to reason about properties of the individual system components’ behaviour and their interaction. This section introduces an informal computational model for Policy-based Runtime Tracking System (PRTS). Tracking is related to the progress of state (e.g. Holding in Quiz) but also it ensures that learner’s outcomes are achieved. In traditional learning environments (i.e. Classroom-based), the learners are continuously observed and the action is taken accordingly. However, in e-learning environments the observer becomes the proposed runtime tracker system. Thus our proposed PRTS is independent automated system that is integrable with any given e-learning environments.

Our computational model precisely attempts to give us an overview of the units
Chapter 3. *Computational Model*

of computation on which our work throughout the thesis is based. Our model also defines what meant by communication, timing and termination.

### 3.2.1 Units of Computation

The computational model can be viewed as a collection of entities that comprise of Observer, Learning Activities, Policies and Runtime Monitoring mechanism as depicted in Figure 3.1 where these units communicate and collaborate with each other to achieve the system objectives.

![Computational Model Diagram](image)

**Figure 3.1:** Computational Model.

#### 3.2.1.1 Observer

Observer is an entity in our model that is responsible for tracing the learning activities performed by learners in an e-learning environment. Observer has a collection of functions that undertake activities which capture the occurrence of
events associated with learning activities. These functions are interfaced with learning activities in the e-learning environment, to be triggered periodically according to the occurrence of events of learning activities. Each function is designed for each particular learning activity type where such functions capture event information such as name, timestamps, status, etc., and send this information to observer storage to be used by other components.

3.2.1.2 Learning Activities

Learning activity is considered as first class citizen in our model. It is a collection of objects designed to support learners have a better understanding and fulfil the learning objectives during the learning process.

The Analysis of Learning Activities

This section aims to extract the necessary activities that are performed during the learning process. Learners perform a number of activities during their presence in the e-learning environment; however, not all activities can probably be learning activities. Learning activities are particular activities designed to support learners achieve their learning objectives and qualifications. Thus, learners may not utilize all learning activities that are available during the learning process.

The analysis concentrates on learning activities that can be tracked where the tracking data is used for enforcing learning activities policies at runtime and providing supportive actions in a timely manner. The information tracked is personalized to the learners and relatively is adequate for describing the effectiveness of the learning activities. These activities include assessment, content interaction and communication and cooperation.

Assessment
Assessment is a learning activity that is very important during the learning process. As described in Section 2.4, assessment is the measurement for the achievement acquired in the learning process. Quiz and assignment are the most popular types of assessment which examine different levels of the learners’ understanding, progress and achievements during the learning process. However, tracking learning activities such as quizzes and assignments is necessary during the learning process. Tracking data is valuable because the learners’ performance and progress can be captured from it. In addition, many circumstances that may influence learners’ achievements during learning process can be avoided by noticing them at the early stage from such data. Tracking data for the quiz include quiz id, time spent, data of attempt, total mark, etc. In addition, such data can be used to provide the tutor with difficulties that a learner has faced; for instance, learner was slow doing the quiz and gained bad mark or a delay in completing the quiz. Thus, providing the teacher with such situations in appropriate time aims to increase awareness of teacher about other quizzes the learner will do in the future.

Learner-Content Interaction

A learner interacts with suitable learning contents during the learning process. Tracking learners’ interaction with learning contents is very important because of the useful information provided which identifies learner’s behaviours. This information may include number of attempts, time spent, last access etc. Tracking such information would lead to identify several attributes about the learning process and student progress that require immediate support and assistance. For instance, it is possible to recognise the learning content that has never been visited by a learner by capturing the number of attempts.

Cooperation and Communication

E-learning systems provide communication tools such as discussion forum and chatting room for cooperation between learner-learner and teacher-learner as a
group or peer to peer. Collaboration within these tools is usually used to discuss the learning content and assessment issues. Participation in such tools leads learners to make contribution with other participants and thus attain more knowledge. Teachers encourage their learners to collaborate during learning process by opening discussion in particular issue regarding learning contents or assessments. Thus, observing students’ collaboration during learning process is important too. Tracking data from this learning activity contain discussion topic, participant id, rate of post, etc. Tracking learner cooperation process can identify active and inactive participants and learners who are away from this learning activity. Moreover, such data can reflect learners’ collaboration behaviour during the learning process.

Finally, analysis of learning activities is concerned with core activities performed during learning process and that can be tracked. Tracking learning activities produces valuable information for improvement purpose. This information can be used to support, guide and adapt learners to the learning activities and; thus, improve the learning process. Tracking learning activities process is part of our proposed system and will be described in more details in the next chapter.

3.2.1.3 Policies

ECA-based policy is represented by set of rules that execute specified actions in response to the occurrence of events and a condition holds TRUE. ECA model can be used to represent almost any policy rules especially the management policies. As described previously, the management policies are usually represented in form of IF-Then rules, thus they are specified based on ECA model. Hence, an ECA-based rule can be represented in form of the following syntax:

On Event

If Condition
Do \textit{Action}

The above syntax comprises of three parameters: Event, Condition and Action; so, on occurrence of Event, if the Condition is satisfied, then the Action is to be executed.

In e-learning context, a policy is a set of rules that ensure learning behaviour and monitors learners’ progress in response to learners interactions with different learning activities carried out within an e-learning environment. The policies are specified in term of learning activity type (e.g. Quiz, Assignment, etc.) and specific requirements which a learner should meet during a period of time. Learning activities policy is considered as a management policy that reflect specific requirements in relation to particular management strategies that predefined by stakeholder (e.g. Instructor, Course designer, Educational institution, etc.). Particularly, we mean by requirements those which are specified about learning activities and the way the leaner behave on them. Hence, the learning activities requirements can be represented by set of policies; so, the policies are enforced to satisfy these requirements. Chapter 7 describes the use of such policies in more details.

\subsection{3.2.1.4 Runtime Monitoring Mechanism}

Runtime monitoring mechanism with the utilised entities undertake activities where the computations of these entities recognise the learning behaviour (i.e. Sequence of events), and interpret the policies and influence the behaviour of learning by enforcing ECA-based policies. The computations are fulfilled at runtime which produce supportive actions (i.e. the output of policies enforcement) to the learners in the e-learning environment. The runtime monitoring mechanism that is deployed in our system are comprised of a recognising events mechanism and an enforcement mechanism.
• **Recognising events mechanism** captures learning behaviour of the learners on a specific learning activity. The observable event (i.e. A snapshot of states) is obscure until it is recognised based on the policy’s attributes. The execution starts by obtaining the policy’s attributes and; consequently, capturing the events that fit with the triggered policy to be checked with the relevant policy. These events are already stored in PRTS Events Repository by Observer component. Event Recogniser component undertakes the activities of capturing the events at runtime in our model.

• **Enforcement mechanism** Interprets policies and influence the learning behaviour of the learners by enforcing obligations based on occurrence of the events that are associated with the learning activities. The occurrence of the events specified in a policy is indicated by capturing the events in the recognising event mechanism at runtime. The Checker component in enforcement mechanism is employed with underlying role where the computation results in enforcing policies and influencing the learning behaviour of learners accordingly. Therefore, this process addresses undesired behaviour and produces a supportive action that assists and guides a learner to the right behaviour on performing learning activities in the e-learning environment.

### 3.2.2 Communication

Entities in our model need to communicate with each other during computation to perform their activities with the desired flexibility. Communication is provided for passing the parameters of interest from one entity to another. The purpose of communication is to help in exchanging and transmitting of information and data between these entities. The communication between Observer component and e-learning environment is asynchronous communication based on specified time constrains. In our model, synchronous communication is provided between components in the Runtime Monitoring Mechanism which gives the means of
cooperation and coordination between the entities of these components to achieve their activities.

### 3.2.3 Time

Time is an important issue in PRTS where the aim is to achieve faster execution time of the system components. Each component in our model has its own internal clock. Each component must synchronise its clock periodically with the server clock. A computation has a set of execution time constraints, minimum and maximum execution time for a computation that may be chosen from this set at the implementation level or at runtime. The role of timing model is to maintain and control system time; thus, preventing the execution time from running longer than it is allowed.

### 3.2.4 Termination

The invocation of Observer entity’s functions starts on the occurrence of an event associated with a particular learning activity. However, this execution finishes due to normal termination or failure termination. The execution of entities in the runtime monitoring components are working in sequential and parallel depend upon the tasks to be completed. Thus, the termination of this computation occurs when all tasks are done and due to successful enforcement of policies which indicated be delivering the notification (i.e. The feedback of policy enforcement) to the learners in the e-learning environments.

### 3.3 Tempura

Tempura is an executable subset of Interval Temporal Logic (ITL) and is originally proposed by Ben Moszkowski (Moszkowski 1986, 1985). A Tempura program is
deterministic, i.e. no arbitrary choice (either of computation length or variable assignment) can be made during execution. For instance, neither the formula \( \neg \text{skip} \) nor the formula \((I = 0) \lor (I = 1)\) is executable, as both are non-deterministic. The syntax of Tempura is restricted to exclude formula such as \( \neg \) and \( \lor \). Data types in Tempura are integers, booleans and lists, out of which more complex ones can be built. The roots of Tempura are in functional programming, imperative programming and logic programming paradigms. The author of this thesis is interested in the specification and validation of learning activity policies. The formalism that we require for this purpose has to be dual in the sense that it allows reasoning about learning behaviour (specification aspect) as well a framework to execute them (validation aspect). An important motivating factor for choosing Tempura is that it provides us with sound logical formalism and sound dynamism for our runtime validation mechanism. Tempura can be extended to contain most imperative programming constructs and at the same time retain its distinct temporal feel.

### 3.3.1 The Language: TEMPURA

Tempura is an executable subset of ITL and the syntax of Tempura reflects the relationship. Tempura has state and static variables defined over primitive types such as Integers and Booleans and over derived types such as lists. Lists in Tempura range over the primitive types and over lists themselves. Lists are analogous to Arrays or Vectors in imperative programming languages. Tempura provides standard operations over expressions such as \(+\), \(-\), \(*\), \(/\), \(\text{div}\), \(\text{mod}\); \(=\), \(>\), \(or\), \(and\). Many interesting operators can be further defined over the syntactical constructs. Tempura communicates with external entities using the parametrised input and output functions. Inputs could be read from the keyboard or any external program. Outputs are produced on the terminal, written to a file or streamed to an external program.
Tempura programs include several elementary operators. As usual, “exp” means an arbitrary expression, b stands for a boolean expression and P and P’ stand for programs. They are:

- Equality: exp1 = exp2.
- Parallel composition: P ∧ P’.
- Conditional: if b then P else P’.
- Local variables: ∃ υ : P.
- Termination: empty.
- Next: next P.
- Always P : □ P.
- Sometimes P : ◊ P.
- Sequential composition: P ; P’.
- Iteration: P*.

Variables in Tempura have the same syntax as in ITL. The basic statements (with the corresponding ITL constructs) are presented in Table 3.1:

<table>
<thead>
<tr>
<th>ITL</th>
<th>Tempura</th>
</tr>
</thead>
<tbody>
<tr>
<td>f₁ ∧ f₂</td>
<td>f₁ and f₂</td>
</tr>
<tr>
<td>A := exp</td>
<td>A := exp</td>
</tr>
<tr>
<td>more</td>
<td>more</td>
</tr>
<tr>
<td>empty</td>
<td>empty</td>
</tr>
<tr>
<td>□</td>
<td>always</td>
</tr>
<tr>
<td>◊</td>
<td>sometimes</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>if b then f₁ else f₂</td>
<td>if b then f₁ else f₂</td>
</tr>
<tr>
<td>while b do f</td>
<td>while b do f</td>
</tr>
<tr>
<td>repeat b until f</td>
<td>repeat b until f</td>
</tr>
<tr>
<td>“procedures”</td>
<td>define p(e₁, ..., eₙ) = f</td>
</tr>
<tr>
<td>“function”</td>
<td>define g(e₁, ..., eₙ) = e</td>
</tr>
</tbody>
</table>

Table 3.1: Basic Statements.

The following is a very simple example of Tempura program to demonstrate a “while” loop. The user inputs the initial values of “X”. The value of “X” in
subsequent states is to be decreased by a constant value of 1. If the value of “X” equals 0, then the program will be terminated.

\[
A \text{ sample Tempura program:}
\]
/* testwhile.t*/
/* Test run of While loop*/
define \( whileLoop() = \{ \\
exists X : \{ \\
input (X) \\
and \\
always output (X) \\
and \\
while not (X=0) do \\
\{skip \\
and X:=X-1 \\
\} \} \} \)
The output:
Tempura 10 > load “testwhile”.
run whileLoop(). Reading file testwhile.t

Tempura 11 >
State 0: > X=3
State 0: X=3
State 1: X=2
State 2: X=1
State 3: X=0
Total reductions: 72 (72 successful).
Maximum reduction depth: 9.

The following shows the termination of while loop because the value of
X entered is zero.

Tempura 12 > load “testwhile”.
un whileLoop(). Reading file testwhile.t

Tempura 13> State 0: > X=0

State 0: X=0
Done! Computation length: 0. Total Passes: 1.
Total reductions: 15 (15 successful).
Maximum reduction depth: 8.
3.3.2 The Tool: AnaTempura

An integral part of the executable framework for ITL is a semi-automatic tool, “AnaTempura”. It provides an integrated workbench for the runtime validation of systems using ITL and its executable subset Tempura. AnaTempura provides:

- specification support
- validation support in the form of simulation and runtime testing in conjunction with formal specification.

AnaTempura is built upon the C-Tempura interpreter, originally developed by Roger Hale (Hale 1988, 1987) and is now maintained by Antonio Cau and Ben Moszkowski. The first Tempura interpreter was programmed in Prolog by Ben Moszkowski (Moszkowski 1986) and was operational around December 2, 1983. Subsequently he rewrote the interpreter in Lisp (mid March, 1984). The C-Tempura interpreter was written in early 1985 by Roger Hale at Cambridge University. An overview of the runtime validation process in AnaTempura is depicted in Figure 3.2. AnaTempura is an open architecture that allows new tool components to be easily “plugged-in”. It monitors and analyses reactive and time critical systems.

![Figure 3.2: Runtime Validation Process.](image)

AnaTempura supports the idea of runtime validation of systems. A possible runtime behaviour of a system can be checked against a property for satisfiability. Since an ITL property corresponds to a set of sequences of states (intervals), runtime validation is just checking whether the sequence generated by the system is a member of the set of sequences corresponding to the property wanted to be checked.
Chapter 3. *Computational Model*

The Tempura interpreter is used to do this membership test. A formula in Tempura is executable through AnaTempura if:

- it is deterministic.
- the length of the corresponding interval is known.
- the values of the variables (of the formula) are known throughout the corresponding interval.

The general framework for analysis can be described as follows:

- **Formulate all desirable properties of the system of interest in Tempura.** The starting point is formulating all desired properties of interest for the system in Tempura. Establishing system properties can be hard. An initial guideline for choosing properties could be formulating safety and liveness properties for the system. Properties formulated in Tempura are stored in files and loaded at runtime.

- **Using Tempura, check that the behaviour satisfies the desired properties.** AnaTempura generates a state-by-state analysis of the system behaviour as the computation progresses. At various states of execution, for which assertion points are defined, values for variables of interest are passed from the system to AnaTempura. The Tempura properties are validated against the values received. If the properties are not satisfied AnaTempura will indicate the errors by displaying what is expected and what the current system actually provides. Therefore the approach is not just a ‘keep tracking’ approach (i.e. giving the running results of certain properties of the system). By not only capturing the execution results but also comparing them with formal properties, AnaTempura performs the validation.

Figure 3.3 shows an overview of the AnaTempura Architecture. Inputs to the system are the source code augmented with assertions points or an ITL specification.
and the properties of interest. The output is a result stating the satisfiability of that property for the system. For a more visually appealing result, the process of validation can be animated. The tool can analyse programs written in C, Verilog and Java. AnaTempura can be downloaded from (Cau & Moszkowski 2011) and several examples of the tool in action can be found at (Cau et al. 2002, Moszkowski 1986).

3.4 Summary

In this chapter, a computational model was presented for the design of Policy-based Runtime Tracking System. The units of computation have been described including the Observer, Learning Activity, Policy and Runtime Validation Mechanism. The chapter also has described the notion of Communication, Timing and Termination in relation to the execution of system units. The syntax and notations of Tempura and the use of the runtime validation engine have been reviewed.
Chapter 4

Architecture

Objectives

- *Provide an overview of our system architecture.*

- *Illustrate our architecture layers and components.*

- *Highlight the information types about learning activities that can be tracked by Tracking Layer.*

- *Provide a detailed description of the functionalities of each component.*


Chapter 4. Architecture

4.1 Introduction

This chapter aims to provide an overview of our proposed architecture of PRTS. Thus, Section 4.2 describes the main aim of PRTS architecture and introduces the architectures layers including: Tracking Layer, Runtime Validation Layer and Action Layer. Each layer is described including its components and functionalities. Particularly, Section 4.3 provides an overview on the e-learning environment (MOODLE). Section 4.4 Describes the Tracking layer including highlighting the information types of learning activities that can be tracked from the e-learning environment, the functionality of the Observer component and the design of the Event Repository which includes the tables that hold the tracked data that are captured by the Observer components. Then, Section 4.5 gives a detailed description of the Runtime Validation layer including its components. Finally, Section 4.6 presents a description of the Action layer and the functionalities of its components.

4.2 Overview of The Architecture

The tracking is a fundamental concept in education because without providing tracking mechanism, learners may face difficulties in achieving the learning objectives. Tracking mechanism should be able to meet the learners’ needs and fulfil the stakeholder requirements by providing supportive feedback in a timely manner. Tracking is an important tool that is fixed within e-learning environments. Most of current VLEs (e.g. MOODLE) are utilized with similar tracking functionalities which aim at recording statistical data for each learner including the number of times, time, date, frequency and IP address of each student accessed course contents, discussion forums, course assessments, and assignments.

The current e-learning environments are unable to track individual learning activities where the tracked information can be used to support and guide learners.
Learners can not receive supportive feedback about their learning process. Current tracking tool do not consider the requirements of stakeholders (e.g. Instructor, tutor, educational institution). Policy-based approach can be employed to define the way the learner behave on single learning activity where the policy enforcement satisfies such requirements. Tracking individual learning activities can provide useful information that can be used to enforce learning activity policy at runtime; thus, provide supportive feedback.

We propose a policy-based runtime tracking architecture for e-learning environments. Our approach is to track and observe the learning activities and the learners’ interactions with them based on a set of policies to provide the learners with a supportive feedback accordingly. The term “Policy-based” brings attention to that the policies play a major role in the tracking mechanism within our proposed approach. The policies are specified in term of learning activity type (e.g. Quiz, Assignment, etc.) and specific requirements which learner must meet during a period of time. Learning activities policy is considered as a management policy that reflect specific requirements in relation to particular management strategies that predefined by stakeholder (e.g. Instructor, Course designer, Educational institution, etc.). Therefore, the key aspects of our approach are:

- How to capturer the learners’ interactions with the learning activities.
- How to formalise and enforce the predefined learning activities policies on captured learners’ interactions.
- How to provide a supportive action in timely manner to learners that guide the learners and support the learners’ progress during learning process.

As shown in (Figure 4.1), the architecture is divided into three layers: Tracking layer, Runtime Validation layer and Action layer.
4.3 E-learning Environment

We have chosen MOODLE (Modular Object-Oriented Dynamic Learning Environment) as a reference learning environment for building our prototype system. MOODLE is a VLE that let instructors provide courses and learning activities, and course management system (CMS) which is designed to assist educational institutions that tend to make quality online courses (moodle.org 2012, Zenha-Rela & Carvalho 2006). The reason of choosing MOODLE is because it is a free open source e-learning product and widely adopted by universities and educational institutions around the world (Zenha-Rela & Carvalho 2006, Cole & Foster 2012).
4.4 Tracking Layer

This layer aims to monitor and observe learning activities that are performed by learners. It has one component called Observer (see Figure 4.1) that detects and captures continuously and concurrently the data about learners’ interactions with the learning activities (such as quiz, assignment, attempts to access the course and so on) performed by the learner in the e-learning environment. The tracking data collected by the Observer component is stored in the Event Repository. The Event Repository consists of tables that hold the information of learning activities and the interactions data that learners left behind with the learning activities. Moreover, they hold information of the learners and the course which the observed learning activities are associated with. The Tracking Layer in our architecture is designed to ensure that each event which is associated with a learning activity and represents the interactions made by a learner is detected by the observer component once it occurred and stored in the Event Repository. The main functionality of the Observer component in this layer is to detect and collect any possible data about learning activities and interactions. The following sections provide a detailed description of the Tracking Layer.

4.4.1 Tracking Learning Activities

The term “Tracking” in relation to e-learning context is the observation process of any possible interaction with learning activities that learners make during the learning process in order to support and assist learners effectively to achieve their learning objectives. This process in Tracking layer in the PRTS architecture (Figure 4.1) consists of capturing any possible information about learning activities that are performed by the learner in the e-learning environment. The tracked
data is stored in a repository for other purposes (e.g. to understand learning behaviour of learners). The Tracking layer is interfaced with e-learning environments. As stated previously in Section 4.3, MOODLE (moodle.org 2012) was chosen in this research as the e-learning environment of our architecture. The collected information from the e-learning environment is related to a user (i.e. enrolled learners and instructor of the course), course and learning activities. The information of learning activities relates to the traces (i.e. refers to the data of actions that are performed on particular learning activity by a learner ) of learner during interaction with learning activities. However, the tracking process in the PRTS considers particular learning activities (as described in Section 3.2.1.2) as the main attention in the learning process. These include content engagement, assignment, quiz, collaborative discussion. In the following subsections, the thesis’s author describes the possible information variables that can be captured from the e-learning environment during the learning process.

4.4.1.1 User Information

This is the information which may be captured and stored by the Tracking Layer about the user during the learning process. This information is used by the PRTS to identify a user who interacts with the e-learning environment and performs a specific learning activity. User information is regularly utilized to allow users gain access to the e-learning system through their login. Table 4.1 below describes user information.
### Chapter 4. Architecture

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>User ID</td>
<td>Used to Identify a specific User who accesses the e-Learning environment.</td>
<td>Integer</td>
</tr>
<tr>
<td>User type</td>
<td>The type of user (Learner, Instructor, etc..).</td>
<td>Text</td>
</tr>
<tr>
<td>First Name</td>
<td>The user’s first name enrolled in particular course.</td>
<td>Text</td>
</tr>
<tr>
<td>Last Name</td>
<td>The user’s last name enrolled in particular course.</td>
<td>Text</td>
</tr>
<tr>
<td>Course ID</td>
<td>Used to identify specific course ID that the learner has been enrolled in, or assigned to instructor.</td>
<td>Integer</td>
</tr>
<tr>
<td>Date of Birth</td>
<td>User’s date of birth.</td>
<td>Date</td>
</tr>
<tr>
<td>Gender</td>
<td>User’s gender.</td>
<td>Text</td>
</tr>
<tr>
<td>Enrolment Date</td>
<td>The date that learner enrolled in a particular course or, the date that the instructor was assigned to the course.</td>
<td>Date</td>
</tr>
</tbody>
</table>

**Table 4.1:** User information details.

### 4.4.1.2 Course Information

The course information is essential in the tracking process as there may be more than one course that is provided to learners. Consequently, this information is useful to capture learning activities information that are linked to each course in the e-learning environments. Table 4.2 depicts the course information’s variables.
Chapter 4. Architecture

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course ID</td>
<td>Used to identify a specific course which contains a number of learning activities.</td>
<td>Integer</td>
</tr>
<tr>
<td>Course Name</td>
<td>The title of the course.</td>
<td>Text</td>
</tr>
<tr>
<td>Description</td>
<td>Brief description about the course and its contents.</td>
<td>Text</td>
</tr>
<tr>
<td>Start Date</td>
<td>The date that the enrolled learners can start accessing the course’s contents and performing its learning activities.</td>
<td>Date</td>
</tr>
</tbody>
</table>

Table 4.2: Course Information details.

### 4.4.1.3 Assignment Information

Assignment is one of the most important learning activity as it is used as a measurement of a learner’s understanding and achievement attained during the learning process. The information related to assignment is decisive and it would be of great importance during the tracking and monitoring process in the PRTS; to understand the learning behaviour. Table 4.3 describes information variables of the assignment learning activity that can be captured by the Tracking Layer.
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assignment ID</td>
<td>Used to Identify a particular assignment that is created for the course.</td>
<td>Integer</td>
</tr>
<tr>
<td>Course ID</td>
<td>This is the course identification number to which the assignment is linked.</td>
<td>Integer</td>
</tr>
<tr>
<td>Name</td>
<td>The name of the assignment that describes what the assignment is about.</td>
<td>Text</td>
</tr>
<tr>
<td>Start Date</td>
<td>The date that the assignment is made available to learners.</td>
<td>Timestamp</td>
</tr>
<tr>
<td>Due Time</td>
<td>The due date and time of the assignment is the last chance for learners to submit this assignment.</td>
<td>Timestamp</td>
</tr>
<tr>
<td>Grade</td>
<td>The total mark which is specified for the assignments.</td>
<td>Integer</td>
</tr>
<tr>
<td>Submission Status</td>
<td>The status of assignment’s submission (Submitted (1)/ Not Submitted (0)).</td>
<td>Boolean</td>
</tr>
<tr>
<td>Submission Time</td>
<td>The date and time when the learner has submitted the assignment.</td>
<td>Timestamp</td>
</tr>
<tr>
<td>User ID</td>
<td>The learner ID who submitted the assignment.</td>
<td>Integer</td>
</tr>
<tr>
<td>Gained Mark</td>
<td>The total mark that the learner was granted.</td>
<td>Float</td>
</tr>
</tbody>
</table>

Table 4.3: Assignment information details.

4.4.1.4 Quiz Information

One of the common assessment tool is Quiz as a learning activity. The information about a quiz is essential and is used to measure the learner achievement and learning behaviour during the learning process. A number of quizzes usually are utilized sequentially during the learning process. Therefore, information related to quiz is valuable to find out the learner’s misunderstanding of a particular part of the course and, to provide the needed assistance to learners in appropriate time. The information variables that may be tracked from this learning activity is described in Table 4.4.
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quiz ID</td>
<td>Used to identify a particular quiz that is created for particular course.</td>
<td>Integer</td>
</tr>
<tr>
<td>Course ID</td>
<td>This is the course identification number to which the quiz is linked.</td>
<td>Integer</td>
</tr>
<tr>
<td>Name</td>
<td>The name of the quiz that describes what the quiz is about.</td>
<td>Text</td>
</tr>
<tr>
<td>Time Open</td>
<td>The time that the quiz is made available to learners.</td>
<td>Timestamp</td>
</tr>
<tr>
<td>Time Close</td>
<td>The time that the quiz is closed.</td>
<td>Timestamp</td>
</tr>
<tr>
<td>Time Limit</td>
<td>The duration that is specified for the quiz to be done.</td>
<td>Integer</td>
</tr>
<tr>
<td>Questions</td>
<td>The number of questions in the quiz.</td>
<td>Integer</td>
</tr>
<tr>
<td>Grade</td>
<td>The total mark that is specified for the quiz.</td>
<td>Integer</td>
</tr>
<tr>
<td>User ID</td>
<td>The learner ID who has done the quiz.</td>
<td>Integer</td>
</tr>
<tr>
<td>Attempts</td>
<td>The number of times that learner attempted the Quiz.</td>
<td>Integer</td>
</tr>
<tr>
<td>Time Start</td>
<td>The time when the quiz was started.</td>
<td>Timestamp</td>
</tr>
<tr>
<td>Time Finish</td>
<td>The time when the learner finished the Quiz.</td>
<td>Timestamp</td>
</tr>
<tr>
<td>Gained Mark</td>
<td>The total mark that the learner was granted.</td>
<td>Float</td>
</tr>
</tbody>
</table>

**Table 4.4: Quiz information details.**

### 4.4.1.5 Resource Information

Resource is the learning activity that is utilised to achieve learning objectives. The resources contain learning contents which learners interact with during the learning process. These include reading materials, video lectures, etc. The Tracking Layer in PRTS can collect information which represents learners’ interaction and engagement with resource learning activity. This information can be used to distinguish a particular resource from the rest that learner interacts with. Moreover, this information is important because it can be used to recognize how learners interact and use these resources during learning process. Table 4.5 shows the information variables details about a resource.
Chapter 4. Architecture

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource ID</td>
<td>Used to identify a particular resource that is added to the course.</td>
<td>Integer</td>
</tr>
<tr>
<td>Course ID</td>
<td>This is the course identification number to which the resource is linked.</td>
<td>Integer</td>
</tr>
<tr>
<td>Name</td>
<td>The name of the resource that indicates what the Resource is about.</td>
<td>Text</td>
</tr>
<tr>
<td>Description</td>
<td>A full description about the resource contents.</td>
<td>Text</td>
</tr>
<tr>
<td>Due date</td>
<td>The date on which the resource due to be attempted.</td>
<td>Timestamp</td>
</tr>
<tr>
<td>User ID</td>
<td>The learner ID who has attempted the resource.</td>
<td>Integer</td>
</tr>
<tr>
<td>Resource Status</td>
<td>The status of the resource (Attempted(1)/ Not attempted(0)).</td>
<td>Boolean</td>
</tr>
<tr>
<td>Attempt Date</td>
<td>The date on which the resource was attempted.</td>
<td>Timestamp</td>
</tr>
<tr>
<td>Attempt Duration</td>
<td>The total time spent on attempting the resource.</td>
<td>Integer</td>
</tr>
</tbody>
</table>

Table 4.5: Resource information details.

4.4.2 Discussion Information

Discussion is one of learning activity that supports learners to attain learning objectives collaboratively during learning process. Some researchers have addressed collaboration learning activity as one of activities that support learning process socially. Information related to discussion is important because it reflects contribution ability and skills of learners. Discussion can be carried out through real-time chatting or discussion forum. Learner can start discussion, participate to existing discussion and view what has been discussed. All these activities are fulfilled to lead to potential improvements during the learning process. Such information is captured by the Tracking Layer in PRTS and more details about information variables of discussion is presented in Table 4.6.
### Table 4.6: Discussion information details.

The information types presented in tables 4.3 - 4.6 are linked to a particular course and related to interactions that take place by each learner during the learning process. There are further information types which may be collected but not considered essential in this thesis. As stated previously, the research work is based on MOODLE and this is effected by the information types described above. However, PRTS can be adapted to support other types of information about learning activities that may be found in other e-learning environments such as Blackboard and WebCT.
4.4.3 Observer Component

This component aims at observing any possible interactions with learning activities that learners make during the learning process (as depicted in Figure 4.2). The described information types in the previous section are observed, captured and stored by this component. The collected information data are utilized in the enforcement of Learning Activity Policies which provide learners with appropriate support in the sense of achieving learning objectives. The functionalities of the Observer component are very important to PRTS because any failure occurs within this component will result in collecting wrong information; thus, affecting other processes in PRTS. The functionalities of this component include checking learners, courses and learning activities information; and tracking new and updated information. These functions are carried out by attaching observation triggers (i.e. SQL triggers) to information records stored in the e-learning environment (as depicted in Figure 4.2).

![Figure 4.2: Observer Component of Tracking Layer.](image)
Chapter 4. *Architecture*

**Checking function** is the initial function utilized within Observer Component. It aims at finding and gathering all information about the following:

- Learners who are enrolled in the courses available within e-learning environment.
- Courses that are available in the e-learning environment.
- All learning activities that are utilized in each course.
- All interactions occurred during learning activities, participation by a particular learner.

This function starts by finding all this initial information in the e-learning environment. Thus, the checking function gathers and stores all found information data safely in the Event Repository to be used by Runtime Validation Layer which is described in the next chapter.

**Observation function** is the main function in the Observer Component. It aims at keeping track of new and updated information continuously during the learning process. This includes collecting any new or updated information related to learners, courses, learning activities or learners’ interactions with learning activities. Observation function are carried out by attaching observation triggers to each information record stored in the e-learning environment data repositories(see Figure 4.2). These triggers are fired once any new inserted data in information record or update made to information data.

The activity diagram in Figure 4.3 shows how courses and learners information are tracked. It starts by activating course and learner triggers which check for new information related to them. If new or updated information found then the triggers are fired to collect these information data and either create and store new data records for them in the Event Repository, or update the existing data records with the updated information.
Tracking interactions with learning activities made by learners is a complicated process. The Observation function starts this process by activating trigger that is specified for a particular learning activity and responsible for collecting any new or updated information data about interactions made on this learning activity. Therefore, any interaction made by a learner will fire the Observation trigger to collect the information data about this interaction which can be new information such as attempting a quiz for the first time, or updated information such as viewing particular learning material for the third time. Moreover, Observation trigger considers collecting the learner ID which indicate to the ownership of interaction information. It also considers obtaining the course ID to indicate the course that learning activity linked to. Observation function stores this information into the Event Repository. These processes are illustrated in the activity diagram in Figure 4.4.
Figure 4.4: Observation activity diagram of Learning Activities.
4.4.4 Event Repository

The Event Repository in the Tracking Layer is the place where all information that are being gathered and collected by Observer Components, are stored. Event Repository tables hold the information data related to courses, learners, learning activities and interactions with learning activities made by a learner during the learning process. Tables are designed based on the categories of information that are described previously. The purpose of the Event Repository in our architecture is to provide well-structured tables for all specified information types to be stored safely, and to guarantee the minimal redundancy of data.

![ERD of Event Repository tables](image)

**Figure 4.5:** ERD of Event Repository tables.

Figure 4.5 illustrates a snapshot of EDR of tables that are designed in the Event Repository with their relationships to each other. Course – INFO table holds
all information about the courses available in the e-learning environments. Information data related to enrolled learners in each course are held in the table called "Learner – INFO". Information data of assessment learning activities are held in Quiz – INFO and Assignment – INFO tables. However, the events that are captured in interactions made by learners are placed in separate tables such as Quiz – EVENTS and Assignment – EVENTS tables, for normalization purposes. Each table in the Event Repository has a primary key whose value is unique for every row of the table. These tables are related to each other by placing the primary key from one table into the related table as a foreign key.

4.5 Runtime Validation Layer

Runtime Validation Layer is a fundamental part in our system architecture. It aims at checking whether or not the captured learning behaviour satisfies given policies by validating these policies at the runtime. Runtime Validation layer undertakes activities that recognize the learning behaviour (i.e. Sequence of events) of a particular learner, and interpret and influence the behaviour of learning by enforcing ECA-based policies. As a policy reflects a set of requirements about learning activities, the validation is achieved by checking whether or not the learning behaviours meet these requirements.

In addition, a runtime validation technique is used based on AnaTempura validation toolkit (Cau & Moszkowski 2011) to accomplish the above process. It is a tool for runtime validation of safety and timing properties (more details about this tool are in Section 3.3.2. We have three components in this layer as depicted in Figure 4.1. These components are described below.
4.5.1 Event Recogniser

The Event Recogniser aims at finding events that are stored in the Event Repository and required by the learning activities policies. Recognising events process is achieved by looking at policy’s attribute and finding events that trigger a policy. Event is a sequence of states that reflect the learning behaviour of a particular learner. Event can be either an information event or a time-based event; which is associated with learning activities’ interactions that are made by learners and stored in our Event Repository. Information event describes the successful interaction’s attempt with a learning activity; for instance when a learner selects a particular quiz. Time-based event describes the occurrence of an event based on a specified time; such as when reaching the time that less than two days remain to an assignment’s submission due date.

![Event Recogniser](image)

**Figure 4.6:** Event Recogniser component in Runtime Validation Layer.

The Event Recogniser component (see Figure 4.6) has four functions: *Obtain policy’s attributes*, *Find events*, *Obtain events* and *Send events*. The activity diagram in Figure 4.7 illustrates the process in this component.

**Obtain policy’s attributes** function aims at looking at attributes of each learning activity policy that are triggered by the Checker component. The learning
activities policies are ECA-based policies (will be described in Chapter 5), thus this function looks at the event part and condition part of the policy and obtains their attributes. The event is specified in a policy which will be enforced on its occurrence. Attributes of an event are the variables of a specified event in a policy that Event Recogniser uses to find and obtain their values from Event Repository. These attributes include the type of learning activity for which the policy is specified, the learner who interact with the learning activity and the action that responds to the type of interaction (e.g. open a quiz, submit a quiz, etc.). The condition’s attributes are the variables of a specified condition in a policy which the Event Recogniser uses to find and obtain their values from the Event Repository. These values is used by the Checker component to evaluate the policy’s condition (this process will be described in Section 4.5.2). The execution of this function starts when the Checker components call the Events Recogniser. Thus, this function receives the attributes of a policy and pass them as parameters to the Find-Values function for further process.

**Find-Values** function is responsible for capturing the places of values of the policy’s attributes. particularly, After a policy attributes are obtained, this function allocates the tables in the Event Repository where values of the required event’s variables are stored and call Obtain-Events function.

**Obtain Events** function retrieves the values that may represent the occurrence of the specified events. Then this function invokes Send Events function with the values for further process.

**Send Event** function aims at taking the values of the specified event and send them in sequence to the Checker components in form of True or False. The True value indicates the occurrence of event and False value indicates the contrast.
4.5.2 Checker

The Checker component is essential in the Runtime Validation Layer which gives our system architecture of The PRTS, the validation functionalities. This component is responsible for interpreting and influencing the learning behaviour of learners based on the events received from the Event Recogniser component, by enforcing learning activities policies accordingly. The Checker component is the engine where the decision is taken place as a result of policy enforcement, thus the feedback that is related to the decision is produced to the Action component.
Chapter 4. Architecture

This component has four functionalities as depicted in Figure 4.8: (i) it obtains a sequence of events from Event Recogniser components, (ii) it obtains policies of learning activities from their repository, (iii) it evaluates learning activities policies based on the obtained events and (iv) it send feedback to Action component about the decision made.

**Figure 4.8:** Checker component in Runtime Validation Layer.

Checker obtains policies from learning activities policies repository in sequence to be evaluated based on the related events that are being received. Consequently, these policies are being triggered by the obtained events accordingly. Obtain events function is responsible for obtaining events that are being sent by the Event Recogniser component. It also receives the relevant information of the
obtained events such as the all learners and courses identification numbers and so forth. The received events start to fire the triggered policies accordingly as each policy is triggered on the occurrence of particular events. After obtaining the events, the Checker component starts to evaluate the policy condition. In accordance to the events received, the Checker validate whether or not the policy conditions match with the related events; thus, if the condition holds TRUE, then the checker enforces the policy action. The enforcement of policy is the decision result of matching process.

**Figure 4.9:** Activity diagram of Checker component’s functionalities.
Send feedback function aims at sending feedback to the Action components about the decision made which contains the type of action that is specified by the policy. The sent feedback include details about the learner ID number and action type as specified in the policy. For example: let us suppose that the action that is specified in a particular policy is in the following form:

\[
(\text{Action} = (\text{LearnerID}, \text{LA.ID}, \text{Alert}, \text{Msg}))
\]

\text{LearnerID} refers to learner ID number, \text{LA.ID} refers to Learning activity type and its ID number which in this example is Assignment, \text{Alert} is the type of action that should be taken which refers to alerting Learner about Assignment based on the evaluated condition of policy, and \text{Msg} refers to the specified message of the action.

The process of executions of the above functionalities is illustrated by the activity diagram in Figure 4.9.

### 4.5.3 Learning Activity Policy

A policy is a set of rules which reflect particular requirements. The Learning activity policy is an ECA-based policy that defines learning paths of a particular learning activity. It is a set of rules that ensures the learning behaviour and regulates the learner progress during the learning process, (See Chapter 5 for more details). As described previously in Section 3.2.1.3, the policy expresses the requirements about the learning behaviour of learners of a particular learning activity, and is predefined by the stakeholder (e.g. Instructor, Course Designer, Educational institution, etc.).

Learning activities policies are placed in a repository which encapsulates the policies information (see Figure 4.1). This repository is the place where the policies
are stored and can be retrieved by Checker component for evaluation. The repository allows the stakeholder (i.e Instructor, Course designer, Educational institution, etc.) to have access to maintain the policies (e.g. store new policy, modify existing policy).

4.5.4 Components’ Interactions

This section shows how components of the Runtime Validation layer interact with each other to achieve the process objectives. The interaction are usually carried out sequentially within the components. The following sequence diagram in Figure 4.10 shows the interactions and behaviour within the components in the Runtime Validation Layer.

![Sequence diagram of Components’ interactions.](image)

**Figure 4.10:** Sequence diagram of Components’ interactions.

4.6 Action Layer

The Action Layer in our system architecture is responsible for performing a supportive action which is the result of the enforcement of an action that is specified in a policy. This component receives a feedback from the Checker component
which responds to an enforced action of a policy. Particularly, The received feedback contains the type of enforced action that is specified in a particular policy, information about the related learner (or instructor in other cases), learning activity and a message that must be delivered to the target learner. Thus, The Action Layer aims at performing the enforced action and delivering its message. In this thesis, the action is considered to be only in the form of notifications that should be displayed to particular learner with the specified messages. The Action layer has one component called Action component which is interfaced with E-learning environment as depicted in Figure 4.1. This component is described with its functionalities in the following subsection.

4.6.1 Action Component

The Action component is important in our architecture because its functionalities are complementary to the enforcement mechanism that is done in the Runtime Validation Layer. This component is interfaced with the E-learning environment, so it ensures the enforced action is performed successfully on the target learner as specified in the enforced policy. This components has four functionalities as illustrated in Figure 4.1; Obtain Feedback, Analyse Feedback, Generate Notification and Perform Notification.

Obtain Feedback function is responsible for obtaining the feedback that is sent by the Checker component. The received feedback contains information about the decision made by the Checker component about the policy enforcement which includes four parts: the targeted Recipient-IDs (i.e either Learner, Instructor or both ), the learning activity IDs, the action type (e.g. Alert, Email) and the message that is specified (e.g. To explains the undesired learning behaviour).
Analyse Feedback function aims at selecting each part in the received feedback and preparing them to generate the notification. It takes first part of the feedback which is about the action recipients IDs, and gather the relevant information including the name and presence status of the recipients in the E-learning environment. The second part is used by this function to obtain the related information of the learning activity. The third part is important to this function because it indicates the form of the notification, either Alert notification or email notification. Alert notification must be displayed in timely manner while the recipients are logged on the E-learning environment. The Analyse Feedback function takes the message from the received feedback, composes it with the relevant information about recipients and learning activity, and generates the notification message.

Perform Notification function is responsible for delivering the notification to the targeted recipients successfully in the specified form. In the form of Alert notification, this function displays the notification as a pop up message while the user is logging on, otherwise it sends an email to targeted recipients. The sequence diagram in Figure 4.12 illustrates the process and the behaviour of the functionalities above.
4.7 Summary

An overview of the proposed architecture has been illustrated and a detailed description was presented for each layer in the architecture including Tracking Layer, Runtime Validation Layer and Action Layer. Details of each layer including their components and functionalities were provided. The process in each component was illustrated using activity diagrams. Sequence diagrams were used to show the interactions and the behaviour between the components.
Chapter 5

Learning Activity Policy

Objectives

• Provide an overview on how a policy can be represented based on ECA model.

• Describe the Learning Activity Policy.

• Describe how the learning activity policies are specified and expressed in Tempura.
Chapter 5. Learning Activity Policy

5.1 Introduction

A policy as mentioned previously in Section 2.6, is a specification of particular requirements and used to regulate a behaviour of a system. Requirements can be represented by a set of rules in a policy which guides decisions and achieves rational outcomes. Thus, the learning behaviour and learners’ progress in an e-learning context can be regulated and ensured by a set of rules accordingly.

The rest of this chapter is structured as follows: Section 7.2 describes the ECA model on which the policy is based, Section 7.3 provided a detailed description about the learning activity policy and how they can be represented based on ECA model and Section 7.4 describes how the learning activities policies can be specified and expressed in Tempura.

5.2 Policy Representations Based on ECA model

There are three common types of policy which are security, access control and management policy. These categories of policy may be deployed in several sectors such as commercial, financial, educational, military, healthcare and so on. The policies may be classified based on their specification within a system, or the properties which the policy should satisfy. The policies can be classified in relation to the way they behave. Accordingly, the policies can generally be classified into: 1) rules dictating behaviour, 2) rules granting or denying permissions and 3) rules as constraints or parameters.

As stated previously in Section 2.6.2, policies are categorised, in term of their functionality, into Security policy and Management policy. Our approach is concerned with management policies. A management policy is used to specify dynamic adaptable management strategies, so they can be easily modified to change the management approach without affecting the management system. Most of
management policies are specified in form of Condition-Action rules, so they can be easily modified because they simply contain conditions and actions.

An ECA-based policy is represented by a set of rules that execute specified action in response to the occurrence of events and a condition that is TRUE. The ECA model can be used to represent almost any policy rule especially the management policies. As described previously, the management policies are usually represented in the form of IF-Then rules, thus they are specified based on ECA model. Hence, an ECA-based rule can be represented in form of the following syntax:

```
On Event
If Condition
Do Action
```

The above syntax comprises of three parameters: Event, Condition and Action. So, on the occurrence of Event, if the Condition is satisfied, then the Action is to be executed. The enforcement process of ECA-based policies (see Figure 7.1) starts at observing the event that specified in the policy rule and used to trigger the rule conditions. When the event occurs, the rule’s condition will be evaluated, so if the condition is satisfied then one or more actions specified within the policy’s rule will be performed accordingly.

![Figure 5.1: Enforcement process of ECA-based Policy rule.](image-url)
ECA model contains a simple rule representation of if-then statement. However, a complex policy can be represented by ECA model by combining one or more rules together where a condition may represent a set of conditions and an action can also be specified as set of actions. Moreover, one rule’s action can trigger another rule and so on. The next section describes what the learning activities policy are, and illustrates how they are represented based on ECA model.

5.3 Learning Activity Policy

As mentioned above, a policy is a set of rules which reflect particular requirements. In e-learning context, it is a set of rules that ensure learning behaviour and monitors learners’ progress in response to learners’ interactions with different learning activities carried out within an e-learning environment. The policies are specified in terms of learning activity type (e.g. Quiz, Assignment, etc..) and a specific requirements which learner should meet during a period of time. Learning activities are a collection of objects designed to support learners to have a better understanding and fulfil the learning objectives during the learning process. Learning activities are particular activities designed to support learners achieve their learning objectives and qualifications. During the learning process, learners utilize and interact with different learning activities such as assessments, collaboration, communication and so on (for more details, see Section 3.2.1.2).

A learning activity policy is considered to be a management policy that reflect specific requirements in relation to particular management strategies that predefined by a stakeholder (e.g. Instructor, Course designer, Educational institution, etc.). Particularly, the author means by requirements those which are specified for learning activities and the way the leaner utilize them. Hence, the learning activities requirements can be represented by set of policies, so the policies are enforced to satisfy these requirements. As aforesaid, the tracking is a fundamental
concept in education because without providing a tracking mechanism, learners may face difficulties in achieving the learning objectives. The tracking mechanism should be able to meet the learners’ needs and fulfil the stakeholder requirements by providing supportive feedback in timely manner. To overcome these challenges, using a policy-based enforcement mechanism is proposed where learning activity policies are specified to support and guide learners to achieve the learning objective, thus to satisfy the requirements.

Learning activity policies are dynamic management policies specified based on ECA Model. The ECA model comprises of: event to be checked, condition to be evaluated and action to be executed. In relation to learning activities, an event can be represented as information about learner’s interaction with a specific learning activity (e.g. selecting Quiz X), or time-based event about specified time related to particular learning activity (e.g. two days remaining to the assignment deadline). A condition can be represented as an expression about the learning activity’s status (e.g. Quiz B = Submitted). An action can be represented as process of specified support that must be provided to the learners.

**Example 7.1:** Let us suppose that a C programming module provided within an e-learning environment provides learners with 10 lessons as reading materials. The instructor has defined a requirement “Learners should access each lesson in sequence”, so a policy that represents this requirement can be specified as follows:

“**When Lesson (n) = Selected, IF Lesson(n-1) = Not-Read Then, Notify Learner (You should read Lesson (n-1) first !!)**”.

The event specified in the policy above is *Lesson(n) = Selected*, where the condition of the policy is *Lesson(n-1) = Not-Accessed* and the action is *Notify*. So, when a learner selected (for example) Lesson (3) and has not accessed the lesson...
(2), then the policy’s action above will be executed and the learner will be notified with a feedback guiding him or her to the right lesson.

5.4 Policy Specification in Tempura

The syntax of Tempura resembles that of ITL (See Section 3.3). It has as data-structures Integers and Booleans and list construct to built more complex ones. Tempura offers a means for rapidly developing, testing and analysing suitable ITL specifications. As with ITL, Tempura can be extended to contain most imperative programming features and yet retain its distinct temporal feel. Advantages of Tempura are that both parallel and sequential composition is expressible, and that it can closely resemble well known programming language constructs. In particular, it contains control structures such as the conditional statement IF THEN ELSE and the iteration statement WHILE. Hence, learning activities policy can be expressed in Tempura.

As stated above, learning activity policies are based on ECA model. An ECA-based rule syntax contains three main components; Event, Condition and Action.

An event of learning activity policy’s rule is represented with its attributes in the form of the following:

\[
\text{On-Event} \ (\text{Learner}, \ L\text{-Activity}, \ Event\text{-info})
\]

Where the **On-Event** is the event expression specified in the rule. **Learner** refers to the **Learner ID** who will perform an action on particular learning activity. **L-Activity** is the **learning activity ID** on which the event is associated with. **Event-info** refer to either the action the learner will perform on the learning activity (e.g.
Event-info = Select), or a specified time that related to the learning activity or an action that is performed by learner on the learning activity (e.g. the time that a learner spent on doing a quiz more than 10 minutes). However, the Learner parameter is optional as in some cases the event is about an action performed on a learning activity that does not involve a learner (e.g. When a Quiz is become available to learners), so the author only needs to know the learning activity ID that an event occurred on it.

A condition of learning activity policy’s rule is represented with its attributes as follows:

**Evaluate-Condition** *(Learner, L-Activity, Condition-Info)*

Where *Evaluate-Condition* is also the specified expression of the rule’s condition. This expression contains three parameters where *Learner* refers to learner ID and *L-Activity* indicates the learning activity ID. *Condition-Info* refers to specific state of the specified learning activity, or particular learner behaviour which satisfies the rule’s condition. This state information can be found in the Event Repository.

An action of policy’s rule is represented as operation with its attributes in the form of the following:

**Do-Action** *(User, L-Activity, Action-Type)*

Where *User* refers *User ID* (e.g. Learner, Instructor, etc..) and *L-Activity* refers to particular *Learning Activity ID*. *Action-Type* corresponds to the type of action that will be performed on the User about the learning activity (e.g. *Action-Type = Notify*). Do-Action is the operation name of the rule’s action. The expressions
Chapter 5. *Learning Activity Policy*

**On-Event** and **Evaluate-Condition** must be True which refers to that the event is occurred and the condition is satisfied, so the **Do-Action** operation is to be performed.

Advantages of Tempura are that both parallel and sequential composition is expressible, and that it can closely resemble well known programming language constructs. ECA-based policy of learning activities can be expressed in Tempura as follows:
define Policy.name(L,L.A) =
exists Event , Condition: {
array1(Event,|L|) and
array1(Condition,|L.A|) and
always {
forall i < |L| : {
forall j < |L.A| : {
On-Event ( L[i], L.A[j], E - info, Event[i])
and
if Event[i] = true then {
Evaluate-Condition (L[i], L.A, C - info, Condition[i])
and
if Condition[j] = true then {
Do-Action (L[i], L.A[j], A - Type)
}
}
}
}
}

Where L refers to a learner and L.A to a particular learning activity. A set of Learners can be represented by a list of learners with length of |L| where each learner is assigned a unique identifier form 0 to |L−1|. Therefore, array1(Event,|L|) and array1(Condition,|L|) create a list of events and a list of conditions with length of |L| where each event and condition should be checked for each learner, thus, each event and condition is assigned a unique identifier form 0 to |L − 1|. The keyword exists corresponds to existential quantification ∃ which provides a means of creating locally scoped variables. As the event and the condition are
Chapter 5. Learning Activity Policy

represented by a list, they are created as local variables. A set of Learning Activities can also be represented by a list of learning activities with length of $|L.A|$ where each learning activity is assigned a unique identifier from 0 to $|L.A - 1|$. The keyword define is used to bind a name to a constant value or to define subprograms. Policy.name is the name given to the policy. The keywords always and forall correspond to the temporal modality □ and the universal quantifier ∀, respectively (the reader referred to (Moszkowski 1986) for more details about the temporal operators).

**Example 7.2:** Let us suppose a policy says that “learner should be reminded about the assignment’s submission one day before the deadline if he or she has not submitted it yet”. This policy can be expressed in Tempura as follows:
Chapter 5. Learning Activity Policy

```
define Assignment-Reminder-P(L,A) = {
exists Event, Condition: {
array1(Event, |L|) and
array1(Condition, |L|) and
always {
forall i < |L| : {
forall j < |A| : {
On-Event ( A[j], One−Day−Left, Event[i] )
and
if Event[i] = true then { Assignment = A[j] 
and
Evaluate-Condition ( L[i], Assignment, Not−submitted, Condition[i] )
and
if Condition[i] = true then {
Do-Action ( L[i], L − A[j], Remind )
}
}
}
}
}
}
```

Assignment-Reminder-P is the name given to this policy. On-Event function will check each assignment deadline, however when one day remains to the deadline, the Event[i] becomes True (i.e True refers to that this Event[i] occurred ). Once the Event[i] occurred in one assignment A[j], the Condition[i] will be evaluated for each learner in response to assignment which has not been submitted. If the Condition[i] holds True, then a reminder action will be executed and each learner
who has not submitted the assignment yet will receive a reminder notification about the deadline.

5.5 Summary

In this chapter, a description of policy representation has been provided based on ECA model. The Learning Activity Policy has been defined and described. This chapter also has described the specification of the learning activity policy in Tempura and illustrative example has been provided to show how such a policy can be expressed in Tempura code. The next chapter is concerned with the prototype implementation of the PRTS.
Chapter 6

System Prototype

Objectives

- Design the prototype of the Policy-based Runtime Tracking System.
- Describe the implementation of prototype classes.
- Support the research that is presented in this thesis.
6.1 Introduction

The high level design of the prototype implementation of our proposed approach is represented in this chapter. Particularly, Section 6.2 provides an overview on the e-learning environment (MOODLE) which our prototype system is interfaced with. In Section 6.3 the SQL-trigger technique is described as it is used in our prototype implementation, hence a justification of choosing such technique is also highlighted. The structure of the prototype is represented in Section 6.4. Finally, Section 6.5 represents a class diagram showing the relationship between the classes of the prototype implementation.

6.2 E-learning Environment: MOODLE

Our system architecture is interfaced with an e-learning environment to achieve its objectives. As aforesaid, the MOODLE e-learning environment was chosen in our prototype because it is an open source software which gives the ability to download, use and modify it under the terms of GNU. Moodle runs without modification on Unix, Linux, Windows, Mac OS X, Netware and any other system that supports PHP, including most web host providers. Data can be stored in a single database (MySQL and PostgreSQL), but it can also be used with Oracle, Access, Interbase, ODBC and others (moodle.org 2012). Moodle is a CMS and VLE and lets instructors provide courses to learners with appropriate contents and learning activities such as assignments, quizzes, discussion forums and so forth. Moodle stores all information about learners, courses and learning activities in its database (see Figure A.1). All learners’ interactions with learning activities are stored in Moodle’s database. Thus, our prototype only deals with the Moodle’s database in which the all information required information that described in Chapter 4 can be found.
6.3 SQL Trigger Technique

An SQL Trigger is an object of a database that is attached to a specific table (Groff & Weinberg 2010, Churcher 2008). The SQL Trigger technique is database independent. A trigger is an action that takes place at specific time (e.g. After inserting new record). A trigger checks the specified conditions and executes the specified action when an event occurs (i.e. Insert, Update or Delete event). This technique is useful for observing a group of tables or single table to collect any new or updated records or even the deletion of records. The syntax of an SQL trigger contains four parts: Time, Event, Rules (or constrains) and Action. The structure of an SQL trigger is illustrated in the following listing (Groff & Weinberg 2010):

```
1 CREATE [or REPLACE ]
2 TRIGGER trigger-name
3 BEFORE [ or AFTER]
4 INSERT [or UPDATE or DELETE]
5 ON table-name
6 FOR EACH ROW
7 BEGIN
8 IF condition 1 THEN
9 ACTION 1
10 IF condition 2 THEN
11 ACTION 2
12 .
13 .
14 .
15 END;
```

Listing 6.1: SQL Trigger Structure

We chose to use the SQL trigger technique in the prototype of the PRTS architecture for the following reasons:

- As stated previously, all information types about interactions with learning activities made by learners are stored in the database of the e-learning environment, thus, using such technique would be easy and efficient to collect all
the required information once it is available in the database of the e-learning environment.

- Using an SQL trigger does not affect the source code of the e-learning system as the PRTS only deals with its database.

- As the MOODLE learning environment is used in this prototype, the system can be adapted to other e-learning environments such as Blackboard and WebCT. Consequently, using the SQL trigger technique makes this process straightforward by attaching the specified triggers to the databases of these e-learning systems.

6.4 Prototype Structure

Our prototype of Policy-based Runtime Tracking System has been structured into seven essential components as illustrated in Figure 6.1. They aims at fulfilling the system requirements of tracking learning activities based on a set of predefined policies and providing the appropriate support to the learner during learning process. As depicted in Figure 6.1, the prototype components implement the PRTS architecture as described in Chapter 4. We must indicate that the Observing-Initiator and SQL-Trigger components together interact only with the Events database, however, the rest of components work and interact with each other, so the components entirely work to achieve our approach objectives as stated in Section 1.4.
The Table 6.1 lists all prototype classes above with their relation to our system components that are proposed in the system architecture shown previously in Figure 4.1 in Chapter 4.

<table>
<thead>
<tr>
<th>Class</th>
<th>Related Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observing Initiator</td>
<td>Observer</td>
</tr>
<tr>
<td>SQL-Trigger</td>
<td>Observer</td>
</tr>
<tr>
<td>Events Database</td>
<td>Events Repository</td>
</tr>
<tr>
<td>Event Recognizer</td>
<td>Event Recognizer</td>
</tr>
<tr>
<td>Event Finder</td>
<td>Event Recognizer</td>
</tr>
<tr>
<td>Checker</td>
<td>Checker</td>
</tr>
<tr>
<td>Actioner</td>
<td>Action Component</td>
</tr>
</tbody>
</table>

Table 6.1: Prototype Components VS. Architecture Components.
6.5 Prototype Class Diagram

We structured each class in terms of attributes and operations where attributes indicate the properties of the class and operations indicate the methods that the class can perform. The class diagram in Figure 6.2 illustrates all classes in our system prototype and the relationships between these classes. The description of these classes including their methods is allocated in Appendix A.
Chapter 6. System Prototype

Figure 6.2: RTT System Prototype Class Diagram

104
6.6 Summary

This chapter has presented the high level design of the prototype implementation of our proposed approach. It started with providing an overview on the MOODLE E-learning environments which our proposed system are interfaced with. The SQL-Trigger technique has been described as it used in our prototype implementation and a justification of choosing such technique has also highlighted. Then it described the prototype components and their attributes and operations. This chapter also illustrated how these components are mapped onto the system architecture’s components. A class diagram showing the relationship between the classes of the implementation was also represented. The source code of the prototype implementation of the classes described in this chapter is allocated in Appendix B. The next Chapter will discuss the evaluation of our approach and show the results of the PRTS implementation.
Chapter 7

Evaluation

Objectives

- Evaluate the implementation of PRTS.
- Present the results of the execution of system components.
- Evaluate the use of policy-based approach in PRTS.
Chapter 7. **Evaluation**

### 7.1 Introduction

This chapter aims at evaluating our approach of PSRT. A case study starts by presenting a scenario of a distance learning course in Section 7.2 with a number of requirements related to the learning activities for the course provided. Expressing the learning activity policies is presented in Section 7.2.1. Different scenarios are presented in Section 7.2.2 to show how our system components work together to enforce the learning activity policies and issue supportive notifications to the learners. Section 7.3 presents what the use of policy-based approach has added to an E-learning system. Finally, Section 7.4 discussed the features of PRTS approach compared to the reviewed tracking tools that were investigated in Section 2.5.3.

### 7.2 Case Study

In this section, the approach of PRTS will be evaluated through a number of scenarios to show how PRTS works and what the results are from the execution of its components.

- **Scenario 7.1**: An educational institution provides a distance learning course of Computing through a virtual learning environment (MOODLE). The course has four modules provided to the distant learners in the first semester throughout 12 weeks. Networking is one of the modules provided which this thesis focuses on (see Figure 7.1). This module is provided with several learning activities including quizzes, assignments, reading materials, discussion forums, etc. The distant learners should participate in and utilise all learning activities to meet the course requirements and achieve the module successfully. In addition, these activities examine the performance and understanding of learners and assist them to acquire all expected outcomes from the module. We focus in this scenario on quizzes and assignments learning activities where there are six quizzes and three assignments.
that are required throughout semester one. However, due to the flexibility of access to the modules and learning objects, the following requirements are carried out by the institution and should be met by learners when they attempt to perform the learning activities:

1. A learner should start with the quizzes as soon as they are available.
2. A learner should not start a quiz or an assignment until the previous one has been finished.
3. A learner cannot submit assignments after the deadline.
4. Each quiz has a limited period of 10 days for completion.

The utilisation and achievement of these learning activities are monitored by a number of learning activity policies based on ECA model (the reader is referred to Chapter 5 for more details) which intend to guide and support the learner to meet the requirements above.

7.2.1 Expressing Learning activity policies in Tempura

The syntax of Tempura resembles that of ITL (See Table 3.1 in Chapter 3). It has as data-structures Integers and Booleans and list construct to build more complex ones. Tempura offers a means for rapidly developing, testing and analysing suitable ITL specifications. As with ITL, Tempura can be extended to contain most imperative programming features and yet retain its distinct temporal feel. Advantages of Tempura are that both parallel and sequential composition is expressible, and that it can closely resemble well known programming language constructs. In particular, it contains control structures such as the conditional statement IF THEN ELSE and the iteration statement WHILE. Hence, learning activities policy are straightforward expressed in Tempura (for more details about Tempura
see Section 3.3).

As stated above, learning activities policies are based on ECA model. An ECA-based rule syntax contains three main components; Event, Condition and Action.

- The requirement (1) above is represented by the following policy:

\[ \text{Policy-1} = \text{“When the quizzes or the assignments are made available to the learners in the e-learning environment, system notifies the learners.”} \]
The proposed policy above is expressed in Tempura and divided into two policies, one for the Assignments and another for Quizzes as follows:

```
define Policy-1-Q (q,L) = {
    exists Event: {
        always {
            On-Event ( q, Created, Event)
            and
            if Event = true then {
                forall i < |L| : {
                    Do-Action (L[i], q, Notify, Msg) }
                }
            }
        }
    }
```

The policy above is expressed in Tempura without condition where the event is (Quiz = Created) and the action is to notify all learners about the quiz. The second policy for the assignment is similar to the policy above and represented as follows:
define Policy-1-A (a,L) ={
 .exists Event: {
     always {
         On-Event ( a, Created, Event)
         and
         if Event = true then {
             forall i < |L| : {
                 Do-Action (L_[i], a, Notify, Msg)
             }
         }
     }
 }

- The requirement (2) above is represented by the following policy:

Policy-2 = "When a learner attempts to do a quiz, if the previous one is not done, the system alerts the learner with the right quiz to do."

The proposed policy above is expressed in Tempura as follows:
Chapter 7. Evaluation

\begin{verbatim}
define Policy-2 (q,p-q,L) ={
  .exists Event, Condition: {
    array1(Event,|L|) and
    array1(Condition,|L|) and
    always {
      forall j < |L| : {
        On-Event ( L[j], q, Select, Event[j])
        and
        if Event[j] = true then {
          Evaluate-Condition (L[j], p - q, Not-Submitted, Condition[j])
          and
          if Condition[j] = true then {
            Do-Action (L[j], p - q, Alert, Msg)
          }
        }
      }
    }
  }
}
\end{verbatim}

Where this policy is defined as Policy-2 and take quiz ID \((q)\) and the previous quiz ID \((p-q)\) and the number of learners \((L)\). The policy represented above has an event \((Select)\) which triggers this policy, the condition intends to evaluate the previous quiz whether or not has already submitted and the action is to alert the learner to do the previous quiz first.

- **Requirements (3) and (4)** are represented by the following policies respectively:
Chapter 7. Evaluation

**Policy-3** = “When there are two days left to the deadline of the assignment, if a learner has not submitted assignment yet, the system reminds the learner about the deadline and the time left.

The proposed policy above is expressed in Tempura as follows:

```plaintext
define Policy-3 (a,L) ={
  exists Event , Condition: {
    array1(Event,|L|) and
    array1(Condition,|L|) and
    always {
      forall j < |L| : {
        On-Event ( a, Two.Days.To.Deadline, Event[j] )
        and
        if Event[j] = true then {
          Evaluate-Condition (L[j], a, Not - Submitted, Condition[j] )
          and
          if Condition[j] = true then {
            Do-Action (L[j], a, Remind, Msg) 
          }
        }
      }
    }
  }
}
```

Where this policy is defined as **Policy-3** and take assignment ID (a) the number of learners (L). The policy represented above has a time-event (Two.Days.To.Deadline) which triggers this policy when there are two days remaining to the submission deadline, the condition intends to evaluate the assignment (a) whether or not has
already submitted by the learner ($L_{(j)}$) and the action is to remind the learner about the submission deadline.

**Policy-4** = “The system alerts the learners who did not complete the quiz if there are three days left to the limited period of 10 days.”

The following is the proposed policy above expressed in Tempura:

```plaintext
define Policy-4 (q,L) ={
exists Event, Condition: {
array1(Event,|L|) and
array1(Condition,|L|) and
always {
forall j < |L| : {
On-Event ( q, 3Days.remains.of.10Days, Event_{(j)})
and
if Event_{(j)} = true then {
Evaluate-Condition (L_{(j)}, q, Not - Completed, Condition_{(j)})
and
if Condition_{(j)} = true then {
Do-Action (L_{(j)}, q, Alert, Msg) }
}
}
}
}
}
```

Where this policy is defined as Policy-4 and take quiz ID ($q$) the number of learners ($L$). The policy represented above has a time-event (3Days.remains.
Chapter 7. Evaluation

of 10Days) which triggers this policy when there are three days remaining to the limited duration of ten days for the quiz to be completed where the condition intends to evaluate the quiz (a) whether or not has already been completed by the learner \( L_{[j]} \) and the action is to alert the learner to complete the quiz \( q \).

7.2.2 Runtime Validation and Supportive Feedback

In this section, a number of scenarios are presented to show how our prototype system works. These scenarios suppose that an event occurred which is either related to a specific learning activity or a learner’s interaction with a particular learning activity. Since the policies above are written in Tempura, a runtime validation using AnaTempura engine (see Section 3.3.2 for more details) is used to interpret the policies above and enforce them at runtime in response to occurred events. The Checker (as described in Chapter 6) is written in tempura and executed by AnaTempura. The Event Recogniser and Actioner components (see Appendix A.4) in prototype system are written in Java. Thus, Java-Tempura Bridge is used as a communication interface between the Tempura interpreter and Event Recogniser and Actioner component written in Java. It provides functions to access Java objects from the Tempura file, enabling event-recognition and supportive feedback to be provided to learners.

- Scenario 7.2: Let us suppose that an instructor has created a new quiz and made it available to the learners. According to the Scenario 7.1 above, learners required to start doing the quizzes once it is available to them. The Policy-1 specified above is triggered in response to the events occurred and associated with Quiz learning activity which is “created”.
Chapter 7. Evaluation

Figure 7.2: Execution of Checker by AnaTempura and The enforcement of Policy-1.
Figure 7.2 illustrates the result from the execution of Checker class which is run by the AnaTempura engine. It starts by reading the policies file which is called “ECA-Policy.t” in which they are stored, then it obtains the initial values (i.e ID numbers) of the course, learners, quizzes and assignments. The Checker reads the policies’ attributes in sequence. According to Scenario 7.2, the Checker reads Policy-1 attributes and then sends them to the Event Recogniser which finds the event that trigger this policy. As shown in Figure 7.2, Event Recogniser found a new quiz with ID= 5 that has been created and sent this event value to Checker.

State 7 shows the enforcements process of Policy-1 which has no condition and therefore the action will be performed to all learners. The screen-shot illustrated in Figure 7.3 presents that a learner receive a notification saying that “the Quiz with ID = 5 is available to do now!!”, so this notification is the result of the enforced action in Policy-1. The feedback delivering process to the learner (as illustrated in Figure 7.3) is performed by the Actioner class as described previously in Chapter 6.

- Scenario 7.3: Let us suppose that a learner (e.g with user ID “std2”) accessed the Networking module page where there is a list of Quizzes (Quiz-1, Quiz-2, ..., Quiz-6) available to do, however, this learner selected Quiz-2, but this learner has not done the Quiz-1 yet. According to the requirements in Scenario 7.1 above, this student should do the Quiz-1 first before going through Quiz-2. Policy-2 specified above is triggered to the event that refers to selection of a quiz by a learner.

The event that indicates the Quiz-2 is selected and is recognized, thus, Policy-2 is triggered as the specified event is “quiz = selected”. However, the learner in Scenario 7.3 is selecting Quiz-2 and has not done the quiz-1 yet, therefore the condition specified within Policy-2 holds TRUE as “Quiz-1 = Not Submitted” which results the action specified in this policy is to be executed. The screen-shot as depicted in Figure 7.4 illustrates the notification feedback which is displayed to the learner std2 guiding him to the rights quiz which is in this case is Quiz-1
Figure 7.3: Delivering of the supportive feedback in response to the enforcement of Policy-1.
(note that the number 5 that appears within the feedback in Figure 7.4 refers to the Quiz ID). This is the output of the enforcement of Policy-2 above.

Figure 7.4: The feedback which is resulted from the enforcement of the action in Policy-2.
Chapter 7. Evaluation

If the learner in the last scenario has already submitted the Quiz-1, then the Policy-2 will be triggered as the event is Quiz-2= Selected. However, the Condition will hold False because the “Quiz-1= Not-Submitted” in the condition is not True. Therefore the action will not be executed and there will no feedback displayed to the learner. The output from AnaTempura as presented in Listing 7.1 shows the process of the above which can be seen in State 8 (line 40 to 46).

```
[Reading file C:/tempura-2.16/PRTS/Activity_Policy/checker.t]
[Reading file C:/tempura-2.16/PRTS/Activity_Policy/../library/exprog.t]
[Reading file C:/tempura-2.16/PRTS/Activity_Policy/../library/conversion.t]
[Reading file C:/tempura-2.16/PRTS/Activity_Policy/../Activity_Policy/ECA_Policy.t]
Tempura 4%
State 0: Cfields=1
State 0: CourseIDs=[2]
State 0: --------------------------------------
State 1: Lfields=3
State 1: LearnerIDs=[3,4,5]
State 1: --------------------------------------
State 2: Tfields=1
State 2: TeacherIDs=[6]
State 2: --------------------------------------
State 3: Qfields=3
State 3: QuizIDs=[5,6,7]
State 3: --------------------------------------
State 4: Afields=1
State 4: AssignIDs=[2]
State 4: --------------------------------------
State 5: --------------------------------------
State 6: CourseIDs=[2]
State 6: LearnerIDs=[3,4,5]
State 6: TeacherIDs=[6]
State 6: QuizIDs=[5,6,7]
State 6: AssignIDs=[2]
State 6: (|AssignIDs| - 1)=0
State 6: --------------------------------------
State 7: Event=false
State 7: "No new quiz available"="No new quiz available"
State 7: --------------------------------------
State 7: Event=false
```

120
Chapter 7. Evaluation

State 7: "No new quiz available"="No new quiz available"
State 7: --------------------------------------
State 7: Event=false
State 7: "No new quiz available"="No new quiz available"
State 7: --------------------------------------
State 7: --------------------------------------
State 8: --------------------------------------
State 8: S=5
State 8: O=6
State 8: A=1
State 8: Event=true
State 8: s[j]=5
State 8: Condition=false
State 8: "you have selected the right quiz"="you have selected the right quiz"
State 8: --------------------------------------
State 8: a=2
State 8: C=false
State 9: student 3 has already done the assignment 2
State 9: --------------------------------------
State 9: student 4 has already done the assignment 2
State 9: --------------------------------------
State 9: Condition=true
State 9: Warning !! : StudentID ,5 The deadline for AssignmentID: 2 is very soon !!

LISTING 7.1: Output of AnaTempura when the action of policy is not executed as the Condition equals to FALSE.

- Scenario 7.4: Let us suppose that the Assignment (Assign-2) is due in 12th of Aug 2012, where there are only two days remaining to the deadline. A learner (std3) has not even think to do the assignments as he is not sure about the deadline of this assignment. The Policy-3 in Scenario 9.1 above is triggered by the event specified as “Time Now = Deadline - 2 days ”.

In Figure 7.5, a supportive feedback is displayed as notification output from the enforcement result of the Policy-3. The learner in last scenario has not submitted the assignment-2 yet, thus, the condition specified in the policy holds TRUE which results in executing the action specified in Policy-3. This action is then performed in form of feedback saying that “Warning !! You haven't submitted
Figure 7.5: A supportive feedback displaying in response to the enforcement of Policy-3.
Chapter 7. Evaluation

Assignment-2 Yet !!!, however, the deadline is very soon ” and which is displayed to the learner once login into the Moodle system. This process is done by the Actioner class which receives the feedback from The Checker about the enforced action. So, the feedback of the enforced action in Policy-3 is sent to the Actioner class in form of: (Learner ID, Assignment ID, Remind, “The deadline for Assignment is very soon”). Thus, The Actioner displays this reminder to the target learner Learner ID with the message specified in the enforced action about the target assignment (Assignment ID) as shown in Figure 7.5.

7.3 What The Policy-based Approach Has Added

As part of the evaluation for our approach, this thesis should presents what the use of policy-based approach has added to the tracking process in e-learning environments. From managerial prospective, using a policy-based approach aims at reducing the complexity of a system’s management. Therefore, the integration of a policy-based approach to our system architecture has added the following features:

- The ability of defining standard policies: defining learning activities policies allows our system to choose the rules which defines the captured behaviour of learning. Defining the policies of learning activities in form of rules based on the ECA model helps reducing the complication of supporting the learner and guiding them to the learning activities.

- The ability to selectively enforce the desired support: Expressing learning activities policies based on ECA model increases the ability of our system to choose what appropriate action is to be performed, how to and when to enforce a particular action based on predefined events and conditions.
• The ability to configure and modify the policies: policies generally can be
defined and expressed using simple policy models. Learning activities poli-
cies have been represented in the form of ECA-based rules, thus this helps
easily and quickly to configure and modify the policies when an error or
conflict occurs. It is also easily to add new policy when appropriate.

7.4 Discussion

In the relation to the research title, the main aim was to devise means of support-
ing and guiding learners with a supportive feedback in a timely manner through
policy-based runtime tracking system. Many e-learning environments have been
designed with tools to support learning process; however, some of these tools are
efficient and other inefficient. The research has shown how runtime tracking using
policies can be used effectively to support learners during their learning process.
Using policy-based technique, the research work showed how learners’ progress
and learning behaviour can be specified by a set of policies. A lot of data about
learning behaviour is stored in e-learning system repository and not used to en-
hance the learning process. Therefore, enforcing such policies in response to this
data at runtime leads to offer a supportive feedback; thus, prevent certain learn-
ing circumstances from occurring such as failure of attaining particular learning
objectives. If a supportive feedback is provided to learners in appropriate time, it
would support learners during the learning process.

Several tracking tools developed previously and used within e-learning environ-
ments were reviewed in Section 2.5.3. It was found that these tracking tools
faced some challenges and required improvements. These tracking tools designed
to track learners during learning process; however, they lacked personalisation as
the foundation for supporting the learners. Table 2.2 in Chapter 2 has highlighted
that the reviewed tracking tools lacked of policy-based tracking and runtime sup-
portive feedback functionalities. As monitoring learners’ progress and learning
behaviour depends initially on the requirements of stakeholder (e.g. Tutor), using policy-based technique allows such requirements to be specified by a set of policies where enforcing them at runtime leads to offer learners with a supportive feedback in a timely manner. Offering learners with supportive feedback according to the enforcement of the policies and in a proper time, would satisfies the requirements of stakeholder and enhance the learning process.

As presented in Section 7.2, it has shown that PRTS approach is able to track personalised learning activities and support learners in a timely manner by enforcing a set of policies at runtime and in response to learners’ interactions with the learning activities in an e-learning environment (MOODLE). Using policies to show how a number of requirements predefined by the educational institution can be specified by a number of policies. A demonstration of the PRTS has shown how such policies can be enforced at runtime in response the learners interactions and how a supportive feedback can be offered accordingly to the learners guiding them to the desired learning behaviour.

As compared to the reviewed tracking tools currently used within e-learning systems, the PRTS is a unique approach capable of facilitating learners in a timely manner. The runtime tracking based on a set of policies can respond immediately to learners’ interactions with the learning activities by providing a supportive feedback which greatly improves the learning process.

7.5 Summary

The evaluation of the approach of PRTS has been carried out in this chapter. A case study was presented to show how learning activity policies can represent particular requirements predefined by an institution. A scenario of a distance course was presented with a number of requirements related to the learning activities for the course provided, so the learning activity policies have been expressed in Tempura accordingly. Different scenarios have been presented to show how our
system components work together to enforce the learning activity policies and issue supportive notifications to the learners. We have also shown what the use of policy-based approach has added to tracking process in an E-learning environment. Finally, the features of PRTS have been discussed compared to the reviewed tracking tools investigated in Section 2.5.3.
Chapter 8

Conclusion

8.1 Research Summary

The work in this thesis can be summarised as follows:

Chapter 2: An overview of the learning methods, e-learning environments, and learning activities were provided. The tracking process was reviewed in both traditional classroom learning and e-learning. In addition, the development of some existing tracking approaches was investigated, so some challenges were addressed. At the end of the chapter, an overview was given on policies, and the ECA model, which is usually used to represent the management policies, was also presented.

Chapter 3: A computational model for the PRTS has been presented. The computational model described the nature of the PRTS components, the behaviour of such components, and the notion of Communication, Timing, and Termination. Then, an overview of Tempura was provided, along with its tool-kit AnaTempura, which is used as a runtime validation engine to interpret and enforce our policies at runtime.
Chapter 4: A novel architecture for the PRTS was proposed. An overview of our proposed architecture was provided, which contains three layers: Tracking Layer, Runtime Validation Layer, and Action Layer. The next stage was to design the Tracking layer of our PRTS architecture. A detailed description of the information types of the learning activities that can be tracked by this layer was provided. The Tracking layer has one component called “Observer” which uses an activity diagram to show the observation process within this layer. The Event Repository of the Tracking layer was next described; here, a number of tables were designed to hold the tracked data that are captured by the Observer component. Next, the Runtime Validation layer and the Action layer of the PRTS architecture were described. A detailed description of each layer, including their components with functionalities, was provided. The process of each component was illustrated by activity diagrams. Sequence diagrams were used to show the interactions and behaviour between components.

Chapter 5: At this stage, a description of the policy representation based on an ECA model was given. The learning activity policy was defined and described. The specification of the learning activity policies was described in Tempura, and an illustrative examples was provided to show how such policies can be expressed in Tempura.

Chapter 6: The next step was to present a high level design of the prototype implementation of the PRTS. The chapter began by providing an overview on MOODLE, with which our prototype system is interfaced. Then, the prototype classes were designed, and their attributes and operations were described. A class diagram was also illustrated to show the relationship between the classes of our prototype implantation.
Chapter 7: An evaluation of the approach of the PRTS was carried out. A case study was presented to show how learning activity policies can represent particular requirements predefined by an institution. A scenario of a distance learning course was presented with a number of requirements related to the learning activities for the course provided, so the learning activity policies were expressed in Tempura accordingly. In addition, different scenarios were presented to show how our PRTS components work together to enforce the learning activity policies and issue supportive notifications to the learners. The chapter then showed what the use of the policy-based approach has added to the tracking process in an e-learning environment. At the end of the evaluation stage, the author of thesis discussed the features of our proposed approach, compared to the reviewed tracking tools that were investigated in Section 2.5.3.

8.2 Revisiting Original Contributions

The main contribution of this thesis is a novel runtime tracking system based on learning activity policies for observing learning behaviour and progress within an e-learning environment, so the learners can be provided with relevant and timely support. The tracking is achieved by expressing the desired learning behaviour and progress as policies, where such policies are monitored and enforced at runtime and in an appropriate time. This section revisits the five original expected contributions, represented in Section 1.4 as follows:

- C1: In Chapter 3, a computational model has been developed. The computational model has described the nature of the units of computation and defined the behaviour of such units. The units of computation has been described as a collection of entities that comprise of Observer, Learning Activities, Policies and Runtime Monitoring mechanism where these units communicate and collaborate with each other to achieve the system objectives.
Chapter 8. Conclusion

- **C2**: In Chapter 4, an architecture of the PRTS has been designed. It has three layers, including Tracking layer, Runtime Validation layer, and Action layer. The Tracking Layer aims to monitor and observe learning activities that are performed by learners. The Runtime Validation layer aims to check whether or not the captured learning behaviour satisfies given policies by validating these policies at the runtime. The Action layer is responsible for performing a supportive action, which is the result of the enforcement of an action that is specified in a policy.

- **C3**: In Chapter 5, a Learning Activity Policy has been established with respect to the ECA model. A policy was formalised using Tempura, where Tempura operators allow the dynamic changing (i.e. at runtime) of policies. A language for specifying and enforcing learning activity policies was provided; a learning behaviour and the learners’ progress can be tracked, and supportive feedback can be provided with respect to these policies. Representing the policies in the form of ECA-rules helps to easily configure and modify the policies or to add a new policy when appropriate. A description of the policy representation has been provided based on an ECA model.

- **C4**: In Chapter 6, a prototype of the PRTS has been implemented based on MOODLE and AnaTempura. The SQL trigger technique was used to describe how the proposed approach can be interfaced with an e-learning environment.

- **C5**: In Chapter 7, a current e-learning environment (MOODLE) with a set of scenarios have been used to evaluate the proposed approach. The evaluation has shown the validation of the enforcement process of the learning activity policies at runtime. The features of PRTS have been discussed and shown the proposed approach has developed new features that has not considered in the reviewed tracking tools investigated in Section 2.5.3.

The proposed runtime tracking mechanism ensures that the learning behaviour and progress are recognised and influenced by enforcing a set of policies. The
approach presented here provides supportive feedback to learners according to the enforcement of the policies, so the approach ensures the learning behaviour and improves the progress of the learners during the learning process. The approach for the Policy-based Runtime Tracking System (PRTS) was designed, implemented, and evaluated to be an integral part to any given e-learning environment.

8.3 Revisiting the Success Criteria

To answer the research questions that were set in Section 1.6, the Policy-based Runtime Tracking System (PRTS) has been designed, implemented and evaluated throughout this thesis. Using the policy-based technique within our approach allows the learning activity policies to be specified, expressed and enforced to meet the learners’ need and the educational requirements. Enforcing the learning activity policies at runtime using the runtime validation mechanism leads to provide the needed support and help to the learners in a timely manner.

A set of criteria of success has been defined in Section 1.6. These predefined criteria are revisited as follows:

- **The proposed approach should be able to provide the proposed characteristics that are lacking in the other tracking tools.**

Several tracking tools developed previously and used within e-learning environments were reviewed in Section 2.5.3. It was found that these tracking tools faced some challenges and required improvements. These tracking tools were designed to track learners during the learning process; however, they lacked personalisation as the foundation for supporting the learners. Table 2.2 in Chapter 2 has underscored that the reviewed tracking tools lacked policy-based tracking and runtime supportive feedback functionalities. As
monitoring learners’ progress and learning behaviour depends initially on the requirements of the stakeholder (e.g. tutor), using a policy-based technique allows such requirements to be specified by a set of policies where they are enforced at runtime, which provides learners with timely supportive feedback. This support satisfies the stakeholder’s requirements and enhances the learning process. Therefore, the proposed approach has shown that policy-based tracking and runtime supportive feedback functionalities are implemented and evaluated; so, they are covered 100%.

- **The computational model should describe how the policy-based approach can be used in the runtime tracking mechanism.**
  
The computational model described in Chapter 3 has shown that the runtime tracking mechanism with utilised entities undertakes activities where the computations of these entities recognises the learning behaviour (i.e. sequence of events), and interprets the policies and influences the learning behaviour by enforcing ECA-based policies. The computations are fulfilled at runtime which produces supportive actions (i.e. the output of policy enforcement) for the learners in the e-learning environment.

- **The proposed approach should show how the learning activity policy can be formalised in Tempura to allow dynamic changing of policies.**
  
The research has shown how runtime tracking using policies can be used effectively to support learners during their learning process. As described in Chapter 5, the research work showed how learners’ progress and learning behaviour can be specified by a set of policies. A lot of data about learning behaviour is stored in e-learning system repository and not used to enhance the learning process. Therefore, enforcing such policies in response to this data at runtime leads to offer a supportive feedback; thus, this prevents certain learning circumstances from occurring such as failure of attaining particular learning objectives. If supportive feedback is provided to learners
in appropriate time, it would support learners during the learning process.

- **The proposed approach should be able to provide the learners with supportive feedback at runtime during the learning process.**

  The evaluation of our approach, as presented in Chapter 7, has shown that the learners can receive supportive feedback in response to their interactions with the learning activities. A demonstration of the PRTS has shown how such policies can be enforced at runtime in response to the learners’ interactions and how supportive feedback can be offered accordingly to the learners, guiding them to the desired learning behaviour. The runtime tracking is based on a set of policies that respond immediately to learners’ interactions with the learning activities by providing supportive feedback, which greatly improves the learning process.

### 8.4 Future Work

Further development that is most immediately needed in our current prototype implementation is the dynamic policy modification by the stakeholder (e.g. instructor, course designer, educational institution, etc.). Currently, the learning activity policies are expressed in Tempura as a code that is read by the Checker components in our prototype system. The PRTS should allow the stakeholder to easily modify complex learning activity policies.

Our proposed approach has considered particular learning activities during the learning process. Further developments may be needed to allow other learning activities, such as eye motions, to be tracked by the PRTS.

Another direction of future work is the further development of the PRTS to improve the Action component. The action component in the current PRTS can
provide supportive action in form of notifications that are issued and displayed to the learners. Other forms of supportive actions should be considered such as connecting the learner with an agent for further support.
References


URL: http://www.eduspaces.net/mberry/files/520/1528/primarymoodle.pdf


URL: ://www.cse.dmu.ac.uk/STRL/ITL/index.html


References


References


URL: [http://www.saradunn.net/VLEproject](http://www.saradunn.net/VLEproject)


URL: [http://www.tandfonline.com/doi/abs/10.1080/02602930220138606](http://www.tandfonline.com/doi/abs/10.1080/02602930220138606)


URL: [http://dx.doi.org/10.1002/9781118269558.fmatter](http://dx.doi.org/10.1002/9781118269558.fmatter)


URL: [http://books.google.co.uk/books?id=K0lm9Tkvo2MC](http://books.google.co.uk/books?id=K0lm9Tkvo2MC)
References


**URL:** [http://www.editlib.org/p/11600](http://www.editlib.org/p/11600)


URL: http://www.ietf.org/


URL: http://www.jiscinfonet.ac.uk/InfoKits/effective-use-of-VLEs/index3.htm


URL: http://www.tandfonline.com/doi/abs/10.1080/08923649509526885


URL: [http://jhi.sagepub.com/content/4/3-4/184.abstract](http://jhi.sagepub.com/content/4/3-4/184.abstract)


URL: [http://www.tandfonline.com/doi/abs/10.1080/0963818032000083559](http://www.tandfonline.com/doi/abs/10.1080/0963818032000083559)


References


144
References


URL: http://www.tandfonline.com/doi/abs/10.1207/s15427625tcq1404


URL: http://ww.cedma-europe.org/newsletterarticles.pdf

Shu Chien, P. (2009), ‘Teachers’ perceptions in the development of online k-12 courses’, Proceedings of Society for Information Technology & Teacher Education International Conference 2009, pp. 3096–3099. AACE.

URL: http://www.editlib.org/p/31120


URL: http://www.elearnspace.org/Articles/elearningcategories.htm


URL: http://dx.doi.org/10.1016/j.compedu.2007.05.009


URL: http://whatis.techtarget.com/definition/0sid9ci866691,00.html

References


**URL:** http://books.google.co.uk/books?id=W2clAQAAIAAJ


Webb, N. M. (1982), ‘Group composition, group interaction and achievement in cooperative small groups’, *Journal of Educational Psychology*, pp. 475–484. ERIC.


**URL:** http://books.google.co.uk/books?id=1mA5CahEYC8C


**URL:** http://doi.acm.org/10.1145/986213.986216

URL: http://www.editlib.org/p/26016


Appendix A

The Prototype
A.1 MOODLE Database

Figure A.1: Snapshot of Moodle’s database (moodle.org 2012).
Appendix A. The Prototype

A.2 Prototype classes

A.2.1 Observing Initiator

Observing Initiator is implemented by a Java class which is executed once at the first time of the PRTS deployment. This class provides methods that initialise the observation process. This class aims at: (I) gathering the initial information of the current courses that are available in the E-learning system, enrolled learners, learning activities and interactions with learning activities made already by learners, (ii) generating SQL triggers and attaching them to the tables in the database of e-learning system that hold the required data of learning activities as described previously in Section 4.2. Observing Initiator class (As depicted in Figure A.2) provides a number of methods including Get-Init-Info, Generate-Trigger and Setup-Trigger which achieve the above.

![Structure of Observing-Initiator.java](image)

**Figure A.2: Structure of Observing-Initiator.java**

*Get-Init-Info* method is responsible for gathering all initial information from the E-learning system database about the current courses, enrolled learners, available
Appendix A. *The Prototype*

learning activities in each course and interactions’ data that are already made by learners with the learning activities. *Generate-Trigger* methods aims at creating an SQL trigger for each type of information as describe in Section 4.3. This method takes one parameter of the specified trigger and return the generated trigger to the *Set-up-Trigger* method. The method of *Set-up-Trigger* obtains the generated trigger and attach it to the targeted table in the database of the E-learning system.
A.2.2 SQL Trigger

As stated above, a trigger is an action that take place at a specific time based on the occurrence of an event of inserting, updating or deleting of a record in a particular table in a database. The class of Observing-Initiator generates and attaches a set of SQL triggers to the specified tables in the database of the E-learning system. These attached triggers are responsible for observing the tables of interest that contains the required information. They collect any changes once they occur in the table’s records and store them into our Events database. These changes can be either new inserting, updating or deleting. Our SQL triggers are specified based on the desired information types as described in Section 4.3.

```
CREATE TRIGGER 'quiz-new-attempt'
AFTER INSERT ON 'mdl-quiz-attempts'
FOR EACH ROW
BEGIN
  INSERT INTO rttdb.quiz-attempt-events
  SET quizid = new.quiz,
  userid = new.userid,
  timestart = new.timestart,
  timefinish = new.timefinish,
  total-grade = new.sumgrades;
END;
```

Listing A.1: Example of specified SQL Trigger

Listing A.1 shows an example of an SQL trigger that is specified to observe any attempt made by learner on a quiz learning activity. This trigger is attached to the quiz-attempts table in the e-learning system database and it collects any event of inserting a record into the table to be stored into quiz-attempt-events table in our Events Database (see Figure A.3).

A.2.3 Design of Events Database

Event database tables hold the information data related to courses, learners, learning activities and interactions with learning activities made by learner during the
Appendix A. *The Prototype*

learning process. Tables are designed based on the categories of information that are described previously in Section 4.3. The purpose of Event data in our system prototype is to provide well-structured tables for all specified and required information types to be stored safely, and to guarantee the minimal redundancy of data. In this thesis, we only focus on assessment learning activity including assignments and quizzes, therefore we designed our database tables to hold the specified information data about this type of learning activity.

As illustrated in (Figure A.3), we have seven tables including *Students-info*, *teacher-info*, *course-info*, *assignment-events*, *assignment-submission*, *quiz-events* and *quiz-attempt*. These tables hold only the required information that obtained from the database of e-learning environment and collected by specified set of SQL triggers. We use MySQL ([http://www.mysql.com](http://www.mysql.com)) DBMS to design our database as a common open source database management system.
Appendix A. The Prototype

A.2.4 Event Recognizer

Event Recognizer component is implemented in Java and it is run by the Checker component. It is responsible for recognizing the events that are required by learning activities policy to be enforced. The Event-Recognizer class undertakes activities that provide information about learning activities in each course that are available for learners to take in each course, and obtain attributes of learning activities policy to find state information about learner interactions with learning activities.

Event-Recognizer class has three major methods as depicted in Figure A.4 including Get-Info, Send-Info and Get-POLICY-ATTRIBUTES which achieve the above tasks. Get-Info is responsible for retrieving the information about learners, courses and learning activities. The task of Send-Info method is to send the information data that are collected by Get-Info method to the Checker. The methods of Get-POLICY-ATTRIBUTES aims at obtaining the policy attributes. It reads the data input stream which contains data about attributes of the policy sent from the Checker component.

![Event Recognizer Diagram](image)

**Figure A.4:** Structure of Event-Recognizer.java
A.2.5 Events Finder

Events-Finder component is Java class and called by Event-Recognizer class to find the desired events from Events Database. This class is responsible for retrieving the events based on the policy attributes received by Event-Recognizer and write output stream containing state information about the recognized event in form of (Variable, Value, Timestamp). It comprised of two methods which are Recognise-Event and Send-Event. The task of Recognize-event method is to find and retrieve the desired event from Events Database based. It takes three parameters including Learner ID, Learning Activity ID and Event Type that are specified in a policy and return 1 (True) if it found the event and 0 (False) if the event is not found. Send-Event method is also called by Recognize-Event and responsible for writing output stream to the Checker component containing the state information (i.e. Event) about the found events by the previous method. Figure A.5 illustrates the structure of Event-Finder class.

![Figure A.5: Structure of Event-Finder.java](image-url)
Appendix A. The Prototype

A.2.6 Checker

Checker component is implemented in Tempura and run by the runtime validation toolkit “AnaTempura” (for more details about Tempura and AnaTempura see Section 3.3). Checker class is responsible for interpreting and influencing learning activity policy based on states information received about the learning behaviour on a particular learning activity. It matches the received events with the triggered policy, evaluate the policy’s condition and produces feedback (i.e. policy enforcement information). As depicted in Figure A.6, the enforcement mechanism is effectively achieved by a number of methods in this class including Get-init-Info, Send-policy-Event, Get-Event, Check-Policy, Check-Policy-Condition-Status and Enforce-Policy-Action.

![Diagram of Checker.tempura](image)

**Figure A.6:** Structure of Checker.tempura
A.2.7 Actioner

Actioner component is implemented in Java and run by the Checker component. It aims at receiving feedback from the Checker about an enforced action which is caused by the evaluation of the policy’s condition of a particular policy, and issue a notification to the targeted learner. As depicted in Figure A.7, this class is structured into three major methods including Get-Enforcement-feedback, Get-Related-Info and Send-Notification. Get-Enforcements-Feedback method is responsible for receiving input stream containing feedback of the enforced action sent by the Checker and preparing the notification action accordingly. Get-Related-Info method is called by previous method and aims at collecting some related information such as Learner’s name and Learner’s log-on status and returning the values to Get-Enforcement-feedback method. The task of Send-Notification method is to perform the required notification by displaying a pop up notification message to the targeted learner or sending an email, so the type of notification depends on the action type specified in the received feedback and the log-on status of the targeted learner.

![Figure A.7: Structure of Actioner.java](image-url)
Appendix B

Source Code

The following listings present source code of the classes of our prototype implementation. The implementation presents only a prototype to evaluate the functionalities of the PRTS components work to achieve our approach objectives. However, it is not implemented to a level at which it could be readily commercially utilized.

```sql
/* --------------------------------------------------------
* @author Turki Alghamdi (STRL, DMU, UK)
* talgamdi@dmu.ac.uk
-----------------------------------------------------------*/
DROP TRIGGER IF EXISTS 'moodle2.0'. 'new_insert' //
CREATE TRIGGER 'moodle2.0'. 'new_insert' AFTER INSERT ON 'moodle2.0'. 'mdl_role_assignments'
FOR EACH ROW Begin
Declare nfirstname VARCHAR(60);
Declare ncourseid INTEGER;

IF (new.roleid = '3') THEN

SELECT u.firstname into nfirstname
FROM mdl_role AS r, mdl_course AS c, mdl_context AS k, mdl_user AS u
WHERE r.shortname = 'editingteacher'
AND r.id = new.roleid
AND k.contextlevel = 50
AND k.instanceid = c.id
```
AND new.contextid = k.id
AND u.id = new.userid;

SELECT c.id into ncourseid
FROM mdl_role AS r, mdl_course AS c, mdl_context AS k, mdl_user AS u
WHERE r.shortname = 'editingteacher'
AND r.id = new.roleid
AND k.contextlevel =50
AND k.instanceid = c.id
AND new.contextid = k.id
AND u.id = new.userid;

insert into rttdb.teacher_info set
  teacherid = new.userid,
  name = nfirstname,
  courseid = ncourseid;
END IF;

IF (new.roleid = '5') THEN

SELECT u.firstname into nfirstname
FROM mdl_role AS r, mdl_course AS c, mdl_context AS k, mdl_user AS u
WHERE r.shortname = 'student'
AND r.id = new.roleid
AND k.contextlevel =50
AND k.instanceid = c.id
AND new.contextid = k.id
AND u.id = new.userid;

SELECT c.id into ncourseid
FROM mdl_role AS r, mdl_course AS c, mdl_context AS k, mdl_user AS u
WHERE r.shortname = 'student'
AND r.id = new.roleid
AND k.contextlevel =50
AND k.instanceid = c.id
AND new.contextid = k.id
AND u.id = new.userid;

insert into rttdb.students_info set
  stdid = new.userid,
  name = nfirstname,
  courseid = ncourseid;
END IF;
End
//
Listing B.1: The source code of the New learner trigger.sql

```sql
/* --------------------------------------------------------
* @author Turki Alghamdi (STRL, DMU, UK)
* talgamdi@dmu.ac.uk
----------------------------------------------------------*/
DROP TRIGGER IF EXISTS 'moodle2.0'. 'crs_new_insert' //
CREATE TRIGGER 'moodle2.0'. 'crs_new_insert' AFTER INSERT ON 'moodle2.0'. 'mdl_course'
FOR EACH ROW Begin
  insert into rttdb.course_info
  Set id = new.id, timestart = new.startdate, weeks = new.numsections;
END
//

Listing B.2: The source code of the New Course trigger.sql

```sql
/* --------------------------------------------------------
* @author Turki Alghamdi (STRL, DMU, UK)
* talgamdi@dmu.ac.uk
----------------------------------------------------------*/
DROP TRIGGER IF EXISTS 'moodle2.0'. 'newassign_insert' //
CREATE TRIGGER 'moodle2.0'. 'newassign_insert' AFTER INSERT ON 'moodle2.0'. 'mdl_assignment'
FOR EACH ROW Begin
  insert into rttdb.assignment_events
  Set assignid = new.id, name = new.name, courseid = new.course, timecreated = new.timeavailable, duetime = new.timedue, sumgrade = new.grade;
End
//

Listing B.3: The source code of the New Assignment trigger.sql

```sql
DROP TRIGGER IF EXISTS 'moodle2.0'. 'newassign_sbt_insert' //
CREATE TRIGGER 'moodle2.0'. 'newassign_sbt_insert' AFTER INSERT ON 'moodle2.0'. 'mdl_assignment_submissions'
FOR EACH ROW Begin
```
Appendix B. Source Code

```sql
insert into rttodb.assignment_submission
Set assignid = new.assignment,
userid = new.userid,
submit_time= new.timemodified,
total_grade= new.grade;
End

Listing B.4: The source code of the New Assignment-Event trigger.sql

/**************************************************************/
* @author Turki Alghamdi (STRL, DMU, UK)                     *
* talgamdi@dmu.ac.uk                                        *
/**************************************************************/
DROP TRIGGER IF EXISTS 'moodle2.0'.newquiz_insert;//
CREATE TRIGGER 'moodle2.0'.newquiz_insert AFTER INSERT ON 'moodle2.0'.mdl_quiz
FOR EACH ROW Begin
Insert into rttodb.quiz_events
Set quizid= new.id,
courseid = new.course,
name = new.name,
timeopen= new.timeopen,
timeclose= new.timeclose,
timelimit= new.timelimit,
sumgrade= new.sumgrades;
End

Listing B.5: The source code of the New Quiz trigger.sql

/*****************************/
* @author Turki Alghamdi (STRL, DMU, UK)                     *
* talgamdi@dmu.ac.uk                                        *
*****************************/
DROP TRIGGER IF EXISTS 'moodle2.0'.newq_attempt_insert;//
CREATE TRIGGER 'moodle2.0'.newq_attempt_insert AFTER INSERT ON 'moodle2.0'.mdl_quiz_attempts
FOR EACH ROW Begin
insert into rttodb.quiz_attempt
Set quizid = new.quiz,
userid = new.userid,
timestart= new.timestart,
timefinish= new.timefinish,
total_grade= new.sumgrades;
```

161
Listing B.6: The source code of the New Quiz-Event trigger.sql

```java
/* --------------------------------------------------------
* @author Turki Alghamdi (STRL, DMU, UK)
* talgamdi@dmu.ac.uk
* -----------------------------------------------------------*/
import java.io.*;
import java.lang.*;
import java.util.Calendar;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.SimpleDateFormat;
public class EventRecognizer {
    public static void main(String[] args) throws SQLException, IOException {
        find_C_Info();
        find_L_Info();
        find_T_Info();
        find_Q_Info();
        find_A_Info();
        EventFinder.find_event();
    }
    public static void find_C_Info() {
        int x=0; int y=0;
        String statment = "SELECT 'courseid' FROM rttdb.course_info" ;
        ResultSet re=null;
        try {
            re= DBHelper.executeSql(statment);
            re.last();
            y= re.getRow();
            System.out.println("! PROG : assert C :"+ y +":!
            re.beforeFirst();
            while(re.next())
            {
                x= re.getInt(1);
                System.out.println("! PROG: assert X:"+ x +":! \n");
            }
        } catch (SQLException e) {
            // TODO Auto-generated catch block
        } catch (IOException e) {
            // TODO Auto-generated catch block
        }
```
Appendix B. **Source Code**

```java
private static void find_L_Info () {
    int x = 0; int y = 0;
    String statment = "SELECT `stdid` FROM rttldb.students_info";
    ResultSet re = null;
    try {
        re = DBHelper.executeSql(statment);
        re.last();
        y = re.getRow();
        System.out.println("! PROG : assert L:"+ y +":!
        re.beforeFirst();
        while (re.next())
        {
            x = re.getInt(1);
            System.out.println("! PROG: assert X:"+ x +":!
        }
    }
    catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    finally
    {
        try
        {
            DBHelper.disconnect();
        } // end try
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
```
public static void find_T_Info() {
    int x=0; int y=0;
    String statment = "SELECT 'teacherid' FROM rttdb.teacher_info";
    ResultSet re=null;
    try {
        re= DBHelper.executeSql(statment);
        re.last();
        y= re.getRow();
        System.out.println("! PROG : assert T :"+ y +":!
        re.beforeFirst();
        while(re.next())
        {
            x= re.getInt(1);
            System.out.println("! PROG : assert X :"+ x +":!
        }
    }
    catch (SQLException e) {
        //TODO Auto-generated catch block
        e.printStackTrace();
    }
    finally
    {
        try
        {
            DBHelper.disconnect();
        }//end try
    }
}

public static void find_Q_Info() {
    int x=0; int y=0;
    String statment = "SELECT 'quizid' FROM rttdb.quiz_events";
    ResultSet re=null;
    try {
        re= DBHelper.executeSql(statment);
    }
Appendix B. Source Code

```java
re.last();
y = re.getRow();
System.out.println("! PROG : assert Q:"+y+:!\n");
re.beforeFirst();
while (re.next())
{
    x = re.getInt(1);
    System.out.println("! PROG : assert X:"+x+:!\n");
}
}
catch (SQLException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
finally
{
    try
    {
        DBHelper.disconnect();
    } //end try
    catch (Exception e)
    {
        e.printStackTrace();
    }
}

public static void find_A_Info() {
    int x=0; int y=0;
    String statment = "SELECT `assignid` FROM rttdb.assignment_events" ;
    ResultSet re=null;
    try {
        re = DBHelper.executeSql(statment);
        re.last();
y = re.getRow();

        System.out.println("! PROG: assert A:"+y+:!\n");
        re.beforeFirst();
        while (re.next())
        {
            x = re.getInt(1);
            System.out.println("! PROG: assert X:"+x+:!\n");
        }
    }
    catch (SQLException e) {
        // TODO Auto-generated catch block
    }
} 
```

165
Appendix B. Source Code

```java
//e.printStackTrace();
}
finally
{
    try
    {
        DBHelper.disconnect();
    }//end try
    catch (Exception e)
    {
        e.printStackTrace();
    }
}
}

Listing B.7: The source code of the Event-Recogniser.java

/* --------------------------------------------------------
 * @author Turki Alghamdi (STRL, DMU, UK)
 * talgamdi@dmu.ac.uk
 *----------------------------------------------------------*/
public class Event-Finder {
    public static void find_event() throws IOException, SQLException {
        BufferedReader reader = new BufferedReader
            (new InputStreamReader(System.in));
        System.out.println("Enter : ");
        String x = reader.readLine();
        int s = Integer.parseInt(x);
        if (s != 111) {
            System.out.println("Enter : ");
            String y = reader.readLine();
            int o = Integer.parseInt(y);
            System.out.println("Enter : ");
            String z = reader.readLine();
            int a = Integer.parseInt(z);
            recognize_event(s, o, a);
        } else {
            Action_Performer.findAttribute();
        }
    }
}

166
public static void recognize_event(int SID, int AID, int Action) throws SQLException {
    int x = 0; int y = 0;
    ResultSet re = null;
    String statement;
    switch (Action) {
        case 0: /* Case id refer to type of action we want to check that performed on Activity ID*/
            statement = "SELECT 'timeclose', 'timeopen' " + " FROM rtt_db.quiz_events WHERE (quizid = " + AID + ")";
            try {
                re = DBHelper.executeSql(statment);
                if (re.isBeforeFirst())
                    {
                        while (re.next())
                        {
                            x = re.getInt(1);
                            y = re.getInt(2);
                            long dateNow;
                            Date now = new Date();
                            dateNow = now.getTime() / 1000;
                            int current = (int) dateNow;
                            int days = ((current - y) / 60 / 60 / 24); /* To compare between two timestamp (unix time) and get the number of days.*/
                            int currentDays = ((x - current) / 60 / 60 / 24);
                            System.out.println("days = " + days + " \n");
                            System.out.println("currentDays = " + currentDays + " \n");
                            int result;
                            if (days < 1)
                                {
                                    result = 1; // 1 refer to True !!
                                    System.out.println("!PROG: assert X:" + result + "!:\n");
                                }
                            else
                                {
                                    result = 0; // 1 refer to false !!
                                    System.out.println("!PROG: assert X:" + result + "!:\n");
                                }
                        }
                    } else {
                        System.out.println("!PROG: assert X:" + 0 + "!:\n");
                    }
    }
Appendix B. **Source Code**

```java
} } } catch (SQLException e) { // TODO Auto-generated catch block e.printStackTrace(); System.out.println("! PROG: assert X:"+0+:!\n"); } } try { DBHelper.disconnect(); find_event(); } catch (IOException e) { // TODO Auto-generated catch block e.printStackTrace(); } break; case 1: statement = "SELECT 'timefinish', 'timestart' FROM "+ "rttdb.quiz_attempt WHERE (quizid ="+AID+") AND (userid ="+SID+") "; try { re = DBHelper.executeQuery(statement); if (re.isBeforeFirst()){ while(re.next()){ x = re.getInt(1); y = re.getInt(2); System.out.println("y+ = "+y+" \n"); long dateNow; Date now = new Date(); dateNow = now.getTime()/1000; int current = (int)dateNow; int result; if (x == 0) {
    int seconds = ((current - y)/60); System.out.println("seconds = "+seconds+" \n"); System.out.println("current = "+current+" \n"); } else {
    result = 0;
    System.out.println("! PROG: assert X:"+result+"!\n"); } 
```
Appendix B. Source Code

```java
} } else
{  
    result = 0;
    System.out.println("! PROG: assert X:"+result+"!\n");
}
}
else {System.out.println("! PROG: assert X:"+0+"!\n");}
}
catch (SQLException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
    System.out.println("! PROG: assert X:"+0+"!\n");
}

try {
    DBHelper.disconnect();
    find_event();
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
break;

case 2:  
    statment = 'SELECT 'timefinish' FROM rttdb.quiz_attempt"+
                 ' WHERE (quizid ="+AID++") AND (userid ="+SID++") ";
    try {
        re= DBHelper.executeSql(statment);
        if (re.isBeforeFirst())
            {  
                while (re.next())
                    {  
                        x= re.getInt(1);
                        int result ;
                        if (x != 0)
                            {  
                                result = 0;
                                System.out.println("! PROG: assert X:"+result+"!\n");
                            }
                        else
                            {  
                                result = 1;
                                System.out.println("! PROG: assert X:"+result+"!\n");
                            }
            }
```
try { 
    DBHelper.disconnect();
    find_event();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    break;
    
    case 3: /* Case id refer to type of action we want to check that performed on Activity ID*/
    statment = "SELECT 'duetime' FROM " + "'rttodb.assignment_events WHERE (assignid ="+AID+")" ;
    try {
        re= DBHelper.executeSql(statment);
        if (re.isBeforeFirst())
            {
            while(re.next())
            {
            x= re.getInt(1);
            long dateNow;
            Date now = new Date();
            dateNow = now.getTime()/1000;
            int current= (int)dateNow ;
            int reminesDays= ((x - current ) / 60 / 60 / 24);
            System.out.println("reminesDays = "+reminesDays+ " \n");
            int result ;
            if (reminesDays <= 2 && reminesDays >= 0 )
                {
                result = 1; // 1 refer to True !!
                System.out.println("!PROG: assert X:"+result+":!\n");
            }
            else
                {
                result = 0; // 1 refer to false !!
                System.out.println("!PROG: assert X:"+result+":!\n");
            }
        }
Appendix B. **Source Code**

```java
else {
    System.out.println("!PROG: assert X:="+0+:!\n");
}

} catch (SQLException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
    System.out.println("!PROG: assert X:="+0+:!\n");
}

try {
    DBHelper.disconnect();
    find_event();
    } catch (IOException e) {
        // TODO Auto-generated catch block
    e.printStackTrace();
}

break;

case 4:
    statement = "SELECT 'submit_time' FROM rttdb.assignment_submission" + 
        " WHERE (assignid ="+AID +") AND (userid ="+SID+");

try {
    re= DBHelper.executeSql(statement);
    if (re.isBeforeFirst())
    {
        while (re.next())
        {
            x= re.getInt(1);
            int result;
            if (x != 0)
            {
                result = 0;
                System.out.println("!PROG: assert X:="+result+:!\n");
            }
            else
            {
                result = 1;
                System.out.println("!PROG: assert X:="+result+:!\n");
            }
        }
    }
    else {System.out.println("!PROG: assert X:"+1+:!\n");}
}

} catch (SQLException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```
/* --------------------------------------------------------
 * @author Turki Alghamdi (STRL, DMU, UK)
 * talgamdi@dmu.ac.uk
 * --------------------------------------------------------*/
load "../library/exprog".
load "../library/conversion".
load "../Activity_Policy/ECA_Policy".
/* java Event-Recogniser */
define course = 0.
define learners = 1.
define quizzes = 2.
define assignments = 3.
/***************************/
define array (L,n) = {
    skip and list (L,n) and stable (struct (L))
    and fill (L,n)
}.
define fill (L,n) = {
    exists i :
        chopstar { skip and forall i < n :{
            find_info ("X", L[i])
        } } and always { (stable (L)) }
}.
define find_info (X,Y) = {
    exists T : {

Listing B.8: The source code of the Event-Finder.java
*/
Appendix B. Source Code

```plaintext
get2(T) and
  if avar(T) = X then { Y = strint(aval(T)) }
  else empty
}
).
define fill3(S,Q,L) = {
  L[0] = S and L[1] = Q
}.
define getNrows(X,Y) = {
  exists T:
    get2(T) and
    if avar(T) = X then { Y = strint(aval(T)) }
    else Y = 0
  }
}.
define nCh(X) = { stable(X) }.
/*--------------------------------------------*/
/* run */ define test() = {
  exists CourseIDs, Cfields, LearnerIDs, Lfields,
  TeacherIDs, Tfields, QuizIDs, Qfields, AssignIDs, Afields, StQz:
  { getNrows("C", Cfields) and output(Cfields)
    and array(CourseIDs, Cfields) and output(CourseIDs)
    and format(" ---------------------------\n") and skip }
  { skip and getNrows("L", Lfields) and output(Lfields)
    and array(LearnerIDs, Lfields) and nCh(CourseIDs)
    and output(LearnerIDs)
    and format(" ---------------------------\n") }
  { skip and getNrows("T", Tfields) and output(Tfields)
    and array(TeacherIDs, Tfields) and output(TeacherIDs)
    = nCh(CourseIDs) and nCh(LearnerIDs) and
    format(" ---------------------------\n") }
  { skip and getNrows("Q", Qfields) and output(Qfields)
    and array(QuizIDs, Qfields) and output(QuizIDs)
    and nCh(CourseIDs)
    and nCh(LearnerIDs) and nCh(TeacherIDs)
    and format(" ---------------------------\n") }
  { skip and getNrows("A", Afields) and output(Afields)
    and array(AssignIDs, Afields) and
    output(AssignIDs) and nCh(CourseIDs)
    and nCh(LearnerIDs) and nCh(TeacherIDs)
    and nCh(QuizIDs) and
    format(" ---------------------------\n") }
  { skip and nCh(CourseIDs) and nCh(LearnerIDs) and
    nCh(TeacherIDs) and nCh(QuizIDs) and nCh(AssignIDs)
    and format(" ---------------------------\n") }
```

173
Appendix B. Source Code

Listing B.9: The source code of the Checker.Tempura

/∗ ----------------------------------------∗
* @author Turki Alghamdi (STRL, DMU, UK)
* talgamdi@dmu.ac.uk
* ----------------------------------------∗
define created = 0 .
define select = 1 .
define qSubmit = 2 .
define soon_deadline =3 .
define aSubmit = 4 .
define condition = [true, false].
define onEvent = false.
define notify = 0.
define alert = 1.
define warn = 2.
define bool2int(x) = if x=true then 0 else 1.
//--------------------------------------------------------------
define Policy -1(q,s) = {
exists Event: {
  found_event(q,created,Event) and
  if Event = true then {
    output (Event) and
    (forall j<(|s|) : {
      doAction(s[j],q,notify)
    }
    and format(" --------------------------------------\n")
  }
  else {
    output (Event) and
    output(" No new quiz available") and
    format(" --------------------------------------\n") } }
}

//----------------------------
define Policy -2(q,p_q,s) = {
exists Event,Condition: {
array1(Event,|s|) and array1(Condition,|s|) and
output(q) and output(p_q) and
forall j<(|s|):{
  done_Event(s[j],q,select,Event[j]) and output(s[j]) and
  if Event[j] = true then {
    is_NotDone_Condition(s[j],p_q,qSubmit,Condition[j]) and
    if Condition[j] = true then {
      doAction(s[j],p_q,alert)
    }
    else output(" you have selected the right quiz")
  }
  else output(" No selected quiz found !!")
  and format(" --------------------------------------\n")
}
}

//----------------------------
Appendix B. Source Code

```plaintext
define Policy-3(a,s) = {
    exists Event ,Condition: {

    output(a) and
    deadline_2DaysRemain_event(a, soon_deadline, Event) and
    if Event = true then {
        array1(Condition,|s|) and
        forall j<(|s|):{
            is_NotDone_Condition(s[j], a, isSubmit, Condition[j]) and
            if Condition[j] = true then {
                doAction(s[j], a, warn)
            } else {
                format(" student %t has already done
                        the assignment %t\n", s[j], a)
                format(" --------------------------------------")
            }
        }
    } else {
        format(" No soon deadline was found
                for AssignmentID: %t 
", a)
    }
}.
/*---------------------------------
---*/
define found_event(O,A,E) = {
    exists Z:
        prog_send_ne(0) and prog_send_ne(O)
        and prog_send_ne(A) and
        find_info("X",Z) and
        if Z=1 then (E = true) else (E= false)
}.

define done_Event(S,O,A,E) = {
    exists Z:
        output(S) and output(O) and output(A) and
        prog_send_ne(S) and prog_send_ne(O)
        and prog_send_ne(A) and
        find_info("X",Z) and
        if Z=1 then (E = true and output(E))
        else ( E = false and output(E))
}.

define is_NotDone_Condition(S,O,A,C) = {
    exists Z:
        prog_send_ne(S) and prog_send_ne(O)
        pro
```
and prog_send_ne(A) and
find_info("X",Z) and
if Z = 1 then (C = true and output (C))
else (C = false and output(C))
}
}

Listing B.10: The source code of ECA-Policy.Tempura

```java
/* --------------------------------------------------------
 * @author Turki Alghamdi (STRL, DMU, UK)
 * talgamdi@dmu.ac.uk
 * --------------------------------------------------------*/
import java.io.*;
import java.sql.SQLException;
import java.util.Date;
import javax.swing.*;
import java.awt.event.*;
```
Appendix B. Source Code

```java
public class Actioner {
    public static void findAttribute ()
        throws IOException, SQLException {
        BufferedReader reader = new BufferedReader(
            new InputStreamReader (System.in));
        System.out.println("info :");
        String x = reader.readLine();
        int s = Integer.parseInt(x);
        System.out.println("info :");
        String y = reader.readLine();
        int o = Integer.parseInt(y);
        System.out.println("info :");
        String z = reader.readLine();
        int a = Integer.parseInt(z);
        doAction(s,o,a);
    }

    public static void doAction(int SID, int AID, int Action)
        throws SQLException, IOException {
        String statment;
        String subject;
        String Msg;
        long dateNow;
        Date now = new Date();
        dateNow = now.getTime()/1000;
        int current = (int) dateNow;
        PopupDialogBox dialogBox = new PopupDialogBox();
        switch (Action) {
        case 0: // Case id refer to type of action
            we want to check that performed on Activity ID*/
            subject = (" Warning for QUIZ ID: " + AID + "!!");
            Msg = "Quiz : " + AID + " is available now !!";
            statment = " INSERT INTO moodle.mdl_message " +
                "(useridfrom, useridto, subject, fullmessage, timecreated)" +
                " VALUES (2, "+SID+", "+subject+", "+Msg+", "+current+" )" ;
            try {
                dialogBox.Message(Msg);
                DBHelper.executeQuery(statment);
                DBHelper.disconnect();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
                Events.find_event();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
                Events.find_event();
            }
        }
    }
}
```
Appendix B. Source Code

break;
case 1: /* Case id refer to type of action
we want to check that performed on Activity ID */
    subject = (" Warning for QUIZ ID: " + AID + "!!");
    Msg = "Warning !! You have to do QUIZ : " + AID + " first !!";
    statment = "INSERT INTO moodle.mdl_message " +
                "(useridfrom, useridto, subject, fullmessage, timecreated) " +
                " VALUES (2, " + SID + ", \""+subject+\"\", \""+Msg+\"\", "+current+\")" ;
    try {
        dialogBox.Message(Msg);
        DBHelper.executeQuery(statment);
        DBHelper.disconnect();
        Events.find_event();
    }
    catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
        Events.find_event();
    }
    break;
case 2: /* Case id refer to type of action
we want to check that performed on Activity ID */
    subject = (" Warning for Assignments ID: " + AID + "!!");
    Msg = "Warning !! You Haven't submitted Assignment ID : " + AID + " yet !! , however the deadline is very soon !!";
    statment = "INSERT INTO moodle.mdl_message " +
                "(useridfrom, useridto, subject, fullmessage, timecreated) " +
                " VALUES (2, " + SID + ", \""+subject+\"\", \""+Msg+\"\", "+current+\")" ;
    try {
        dialogBox.Message(Msg);
        DBHelper.executeQuery(statment);
        DBHelper.disconnect();
        Events.find_event();
    }
    catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
        Events.find_event();
    }
    break;
default :
    System.out.println("error!! \n");
    Events.find_event();
Appendix B. Source Code

Listing B.11: The source code of the Actioner.java

```java
import java.sql.Connection;
import java.sql.Statement;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;

public class DBHelper {
    static final String DRIVER = "com.mysql.jdbc.Driver";
    private static final String DATABASE_URL = "jdbc:mysql://localhost:3306/rttdb";
    private static final String USERNABE = "root";
    private static final String PASSWORD = "1234";
    private static Connection connection;
    private static Statement statement;

    private static void connect() throws SQLException, ClassNotFoundException {
        Class.forName(DRIVER);
        connection = DriverManager.getConnection(DATABASE_URL, USERNABE, PASSWORD);
    }

    public static void disconnect() throws SQLException {
        if (!connection.isClosed()) {
            connection.close();
            statement.close();
        }
    }

    public static ResultSet executeSql(String sqlStatement) {
        try {
            connect();
            statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery(sqlStatement);
            return resultSet;
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```
catch (ClassNotFoundException classNotFound) {
    classNotFound.printStackTrace();
}// end catch
    return null;
}

public static void excuteQuery(String sqlstatment) {
    try {
        connect();
        statement = connection.createStatement();
        statement.executeUpdate(sqlstatment);
    } // End Try
    catch (SQLException e) {
        e.printStackTrace();
        } catch (ClassNotFoundException classNotFound) {
    classNotFound.printStackTrace();
}// end catch

LISTING B.12: The source code of the DBHelper.java