AHP-BASED METHODOLOGY FOR A COMPLEX DECISION SUPPORT IN EXTREME PROGRAMMING

A Thesis
Submitted to the Faculty of Graduate Studies and Research
In Partial Fulfillment of the Requirements
For the Degree of
Doctor of Philosophy
in Engineering
University of Regina

By
Sultan Abdullah J Alshehri

Regina, Saskatchewan

Jan, 2014

Copyright 2014: S. Alshehri
Sultan Abdullah J. Alshehri, candidate for the degree of Doctor of Philosophy in Engineering, has presented a thesis titled, *AHP-Based Methodology for a Complex Decision Support in Extreme Programming*, in an oral examination held on December 20, 2013. The following committee members have found the thesis acceptable in form and content, and that the candidate demonstrated satisfactory knowledge of the subject material.

External Examiner: *Dr. Ralph Deters, University of Saskatchewan*

Supervisor: Dr. Luigi Benedicenti, Software Systems Engineering

Committee Member: Dr. Mohamed El-Darieby, Software Systems Engineering

Committee Member: Dr. Amr Henni, Industrial Systems Engineering

Committee Member: **Dr. Malek Mouhoub, Department of Computer Science**

Chair of Defense: Dr. Sandra Zilles, Department of Computer Science

*via SKYPE

**Not present at defense
ABSTRACT

Extreme Programming (XP) is one of the most successful methods in software development. It offers a set of practices designed to work together in order to provide value to the customer. The XP process emphasizes simplicity, feedback, and encouragement among team members. During the XP lifecycle, developers and customers regularly encounter situations in which they need to make decisions or evaluate factors that will affect the development process and team productivity. The Analytic Hierarchy Process (AHP) can analyze complex decisions and help accomplish XP values and fulfill team needs. This research describes a framework for the application of AHP to XP to resolve conflict and evaluate influential factors by structuring the decision making process. AHP seems promising for use in XP for the effective resolution of conflict and the achievement of effective decisions during the software development process.

In this research, more than 10 areas related to XP practices that could benefit from the AHP technique are elaborated upon. Planning game, simple design, metaphor, pair programming, refactoring, and testing are some XP practices whose quality depends on decisions, and thus are ideally suited for AHP. First, in the planning game, two areas were investigated: prioritizing the user stories and ranking the prioritization techniques. Second, AHP explored two areas in the XP simple design practice: selecting the best simple design tool for the XP team and prioritizing the CRC cards as a common simple design tool. Third, in the pair programming practice, AHP was used for two purposes: selecting the best matching pairs and deciding if the pairs should have the same characteristics or be different. Fourth, in the practice of refactoring, AHP was used to
rank the refactoring techniques based on the internal and external quality attributes. Fifth, the AHP was applied to help make decisions about the level of automated testing and ranking the software release indicators.

The AHP is a structure technique based on mathematical models that analyzes various tangible and intangible attributes in order to accomplish a specific goal. It has been applied in numerous fields and many engineering applications. The AHP provides a framework that structures decision problems in order to provide mathematical judgments built on knowledge and experience. AHP has been very useful in software development, where complex decisions occur routinely, and especially useful for unstructured or light processes, like agile processes. The overall results show that in the XP environment, AHP received positive feedback from all participants in terms of the quality of the decisions, team performance and communication, and user satisfaction.
ACKNOWLEDGMENTS

First and foremost I want to thank my supervisor, Dr. Luigi Benedicenti, who helped me with encouragement and patience in all the studying, research, and completing of this thesis. It has been an honor to be one of his Ph.D. students. The joy and enthusiasm towards the research was contagious and motivational for me. I appreciate all his contributions of time, ideas, guidance, wisdom, and support to make my Ph.D. experience productive and stimulating. I am grateful to him for giving me opportunities to have meaningful learning experiences.

My sincerest gratitude goes to my family for all their love and encouragement, especially for my parents, who raised me with a love of science and supported me in all my pursuits. A heartfelt thanks to my wife; without her, it would have been impossible for me to finish this work. I deeply appreciate your love, understanding, and unconditional support.

In addition, I would like to thank Dr. Mohamed Ismail for his collaboration and development efforts.

Many thanks to all my friends, Mohammad Amin, Fateh Hadiby, Intesar Ramely, who played such important roles along the journey as we engaged in various informative discussions and collective work that opened the doors for many solutions that helped me to overcome challenges I have encountered.

Lastly, I would like to thank my internal committee members: Dr. Mohamed El-Darieby, Dr. Amr Henni, and Dr. Malek Mouhoub for their time, interest, and valuable comments.
DEDICATION

To the most inspiring person for my generation in this decade
“Mohamed Bouazizi”
A young Tunisian, who sparked the ongoing Arab Revolutions.
A man, who set himself on fire and died Jan 2011.

Yet, since that moment, we kept singing:
“ If, upon a day, a nation determines to live,
Then fate shall bow down to its decision.
And night shall ebb in submission,
And chains shall weaken and break away.
So he who does not embrace life’s affection,
Shall vaporize in its air and diminish.”

“Abu al-Qasim al-Shabbi”
# TABLE OF CONTENTS

ABSTRACT ....................................................................................................................... i  
ACKNOWLEDGMENTS .................................................................................................... iii  
DEDICATION ................................................................................................................... iv  
TABLE OF CONTENTS ................................................................................................... v  
LIST OF TABLES .............................................................................................................. x  
LIST OF FIGURES ........................................................................................................... xiv  
LIST OF ABBREVIATIONS ............................................................................................. xvi  

## CHAPTER 1: INTRODUCTION

1.1 Research Problem ........................................................................................................ 1  
1.2 Motivation .................................................................................................................... 5  
1.3 Research Objectives ..................................................................................................... 6  
1.4 Methodology and Scope ............................................................................................... 7  
1.5 Original Contributions ................................................................................................. 9  
1.6 Organization of Thesis ................................................................................................ 10  

## CHAPTER 2: LITERATURE REVIEW

2.1 Agile Methodology ........................................................................................................ 12  
2.1.1 Existing agile methods ........................................................................................... 12  
2.2 Extreme Programming ................................................................................................ 13  
2.2.1 Extreme programming process ............................................................................. 13  
2.2.2 Extreme programming values ............................................................................... 15  
2.2.3 Extreme programming practices ........................................................................... 17  
2.3 Analytic Hierarchy Process (AHP) .............................................................................. 20  
2.3.1 Pairwise comparison ............................................................................................. 21  
2.3.2 Priorities derivation ............................................................................................... 23  
2.3.3 Consistency in AHP ............................................................................................... 25  
2.3.4 Advantages of using AHP .................................................................................... 28  
2.3.5 Expert Choice tool ............................................................................................... 29  
2.3.6 Expert Choice main functions .............................................................................. 30  

## CHAPTER 3: SUITABILITY OF AHP TO SOFTWARE DEVELOPMENT PROCESSES

3.1 First Study .................................................................................................................... 33  
3.1.1 Summary and charts for the first study .................................................................. 34  
3.2 Second Study ............................................................................................................... 36  
3.2.1 Summary and charts for the second study ............................................................. 37  
3.3 Third Study ................................................................................................................. 39  
3.3.1 Summary and charts for the third study ................................................................. 40  
3.4 AHP in Software System Applications ...................................................................... 41  
3.4.1 Decision ............................................................................................................... 42  
3.4.2 Assessment measurement ..................................................................................... 46  
3.4.3 Prioritization ......................................................................................................... 48  
3.4.4 Selection ............................................................................................................... 54
CHAPTER 6: APPLYING AHP WITH XP PRACTICES

6.1 Planning Game ........................................ 157
   6.1.1 User stories overview .................................. 157
   6.1.2 Factors affect the requirements prioritization .............. 159
   6.1.3 A Collective activity approach ............................ 164
   6.1.4 Uses of AHP in planning game ............................ 164
   6.1.5 Firstly: Prioritizing the user stories ....................... 164
      6.1.5.1 Domain of study .................................. 164
      6.1.5.2 First finding and results ............................ 167
      6.1.5.3 Second Finding and Results ......................... 174
   6.1.6 Secondly: Ranking the prioritization tools ................. 178
      6.1.6.1 Introduction .................................... 178
         6.1.6.2 Prioritization techniques ......................... 178
         6.1.6.3 Proposed criteria for ranking the prioritization techniques 182
         6.1.6.4 AHP- prioritization techniques structure ............ 182
         6.1.6.5 Pairwise comparison process for the prioritization techniques 183
         6.1.6.6 AHP evaluation results for the prioritization techniques ......... 184
         6.1.6.7 Observations ................................... 186
   6.2 Simple Design ........................................ 189
      6.2.1 Introduction ........................................ 189
      6.2.2 Uses of AHP in simple design .......................... 189
      6.2.3 Firstly: Selecting the simple design tools ................ 190
         6.2.3.1 Proposed criteria for selecting the design tools .......... 192
         6.2.3.2 AHP-simple design structure ........................ 192
         6.2.3.3 Pairwise comparison process for the design tools decision ....... 193
         6.2.3.4 AHP evaluation results for the design tools .......... 194
6.2.3.5 Observations .......................................................... 195
6.2.4 Secondly: Ranking the CRC cards .................................. 197
  6.2.4.1 CRC cards overview ................................................ 197
  6.2.4.2 Proposed criteria for ranking the CRC Cards .................. 198
  6.2.4.3 CRC Cards for the educational case studies ................... 199
  6.2.4.4 AHP-CRC cards for Team 1 ....................................... 200
  6.2.4.5 CRC cards pairwise comparison process ....................... 200
  6.2.4.6 Team 1: AHP evaluation results for the CRC cards .......... 201
  6.2.4.7 Observations .......................................................... 202
  6.2.4.8 AHP-CRC cards for Team 2 ....................................... 204
  6.2.4.9 Team 2: AHP evaluation results for the CRC cards .......... 205
  6.2.4.10 Observations ....................................................... 206
6.3 Pair Programming .......................................................... 208
  6.3.1 Introduction .......................................................... 208
  6.3.2 Research focused in pair programming ........................... 210
  6.3.3 Uses of AHP in the pair programming ............................ 211
  6.3.4 Firstly: Selecting the best pairs .................................. 212
    6.3.4.1 Proposed criteria for selecting the best pairs .......... 212
    6.3.4.2 AHP-best pairs structure ...................................... 213
    6.3.4.3 Pairwise comparison process for selecting pairs ........ 214
    6.3.4.4 AHP evaluation results for selecting pairs ............... 215
    6.3.4.5 Observations (educational case studies) .................. 216
    6.3.4.6 Observations (industrial case studies) .................... 220
  6.3.5 Secondly: Ranking the rules of matching two persons ........ 222
    6.3.5.1 Proposed Criteria for rules of matching two persons .... 222
    6.3.5.2 Rules of pair matching ......................................... 223
    6.3.5.3 AHP-rules of matching structure ............................. 223
    6.3.5.4 Pairwise comparison process for rules of matching ....... 224
    6.3.5.5 AHP evaluation results for rules of matching ........... 225
    6.3.5.6 Observations (educational case studies) .................. 226
    6.3.5.7 Observations (industrial case studies) .................... 230
6.4 Refactoring ................................................................. 232
  6.4.1 Introduction .......................................................... 232
  6.4.2 Refactoring patterns (Techniques) ............................... 235
  6.4.3 AHP uses in refactoring ............................................. 236
  6.4.4 Firstly: Refactoring patterns based on internal quality attributes ... 237
    6.4.4.1 Proposed criteria for ranking the refactoring patterns ...... 240
    6.4.4.2 AHP-refactoring structure for the internal attributes .... 241
    6.4.4.3 Refactoring patterns pairwise comparison process ........ 241
    6.4.4.4 AHP evaluation results for the refactoring patterns ....... 243
    6.4.4.5 Observations (educational case studies) .................. 245
    6.4.4.6 Observations (industrial case studies) .................... 250
  6.4.5 Secondly: Refactoring patterns based on external quality attributes ... 253
    6.4.5.1 Proposed criteria for ranking the refactoring patterns ...... 254
    6.4.5.2 AHP-refactoring structure for the external attributes .... 255
    6.4.5.3 AHP evaluation results for the refactoring pattern ....... 257
CHAPTER 7: CONCLUSIONS AND FUTURE WORK

7.1 Summary .......................................................................................................................... 320
7.2 Future Work ....................................................................................................................... 325
LIST OF REFERENCES ......................................................................................................... 328
LIST OF TABLES

Table 2.1 AHP Numerical scale developed by Saaty.................................................. 22
Table 2.2 Number of comparisons in AHP ................................................................ 23
Table 2.3 AHP criteria in matrix ............................................................................. 23
Table 2.4 Random consistency index for different number of criteria ....................... 28
Table 3.5 AHP articles based on the their themes...................................................... 33
Table 3.6 AHP articles based on the their application areas ..................................... 33
Table 3.7 AHP publications by field......................................................................... 37
Table 3.8 AHP publications based on integrated methods...................................... 37
Table 3.9 AHP applications with its integrated methods.......................................... 40
Table 3.10 AHP purposes in software system development..................................... 83
Table 3.11 AHP and integrated methods in software development.......................... 83
Table 4.12 Comparison among XP studies .............................................................. 102
Table 5.13 Number of hours per iteration for Team 1 and Team 2 ......................... 141
Table 5.14 User stories and tasks for iteration 1 ....................................................... 151
Table 5.15 User stories and tasks for iteration 2 ....................................................... 151
Table 5.16 User stories and tasks for iteration 3 ....................................................... 151
Table 5.17 User stories and tasks for iteration 4 ....................................................... 152
Table 5.18 Technical information per iteration for team 1 ........................................ 152
Table 5.19 Technical information per iteration for team 2 ........................................ 153
Table 5.20 Summary of the industrial projects ......................................................... 154
Table 5.21 Technical information for company A ..................................................... 155
Table 5.22 Technical information for company B ..................................................... 155
Table 5.23 Technical information for company C ..................................................... 155
Table 6.24 Factors that affect the software requirements ........................................ 163
Table 6.25 User stories ranking from the developers............................................... 175
Table 6.26 User stories ranking from the customers............................................... 175
Table 6.27 Ranking the user stories considering each criterion individually ............ 177
Table 6.28 Prioritization techniques ranking for Team 1 .......................................... 185
Table 6.29 Prioritization techniques ranking for Team 2 .......................................... 186
Table 6.30 Prioritization technique based on each criterion individually in Team 1... 187
Table 6.31 Prioritization technique based on each criterion individually in Team 2... 188
Table 6.32 Design tools ranking for Team 1 ................................................................. 194
Table 6.33 Design tools ranking for Team 2 ................................................................. 195
Table 6.34 Design tools based on each criterion individually in Team 1 ................... 196
Table 6.35 Design tools based on each criterion individually in Team 2 ................. 196
Table 6.36 Team 1: CRC1 (Class: Issue) ..................................................................... 199
Table 6.37 Team 1: CRC2 (Class: Assignment) ......................................................... 199
Table 6.38 Team 1: CRC3 (Class: Job) ........................................................................ 199
Table 6.39 Team 1: CRC4 (Class: IssuePriority) ......................................................... 199
Table 6.40 CRC cards ranking for Team 1 ................................................................. 201
Table 6.41 CRC cards based on each criterion individually in Team 1 .................... 202
Table 6.42 Team 2: CRC1 (Class: Authentication) ...................................................... 204
Table 6.43 Team 2: CRC2 (Class: AdminPanel) ......................................................... 204
Table 6.44 Team 2: CRC3 (Class: EditUser) ............................................................... 204
Table 6.45 Team 2: CRC4 (Class: UserForm) ............................................................ 204
Table 6.46 CRC cards ranking for Team 2 ................................................................. 205
Table 6.47 CRC cards based on each criterion individually in Team 2 .................... 206
Table 6.48 Pair ranking for Team 1 ............................................................................. 215
Table 6.49 Pair ranking for Team 2 ............................................................................. 216
Table 6.50 Ranking the pairs by Team 1 based on each criterion individually .......... 217
Table 6.51 Ranking the pairs by Team 2 based on each criterion individually .......... 217
Table 6.52 Pair ranking for company A ..................................................................... 218
Table 6.53 Pair ranking for company B ..................................................................... 219
Table 6.54 Pair ranking for company C ..................................................................... 219
Table 6.55 Ranking the pairs by company (A) based on each criterion individually .... 221
Table 6.56 Ranking the pairs by company (B) based on each criterion individually .... 221
Table 6.57 Ranking the pairs by company (C) based on each criterion individually .... 221
Table 6.58 Ranking the rules of pairing for Team 1 .................................................. 225
Table 6.59 Ranking the rules of pairing for Team 2 .................................................. 226
Table 6.60 Ranking the rules of pairing by Team 1 based on each criterion individually ... 227
Table 6.61 Ranking the rules of pairing by Team 2 based on each criterion individually ... 227
Table 6.62 Ranking the rules of pairing for company A .............................................. 228
Table 6.63 Ranking the rules of pairing for company B .............................................. 229
Table 6.64 Ranking the rules of pairing for company C .............................................. 229
Table 6.65 Rules of pairing by company (A) based on each criterion individually ........ 231
Table 6.66 Rules of pairing by company (B) based on each criterion individually .......... 231
Table 6.67 Rules of pairing by company (C) based on each criterion individually .......... 231
Table 6.68 Ranking the refactoring techniques by Team 1 (internal attributes) .......... 244
Table 6.69 Ranking the refactoring techniques by Team 2 (internal attributes) .......... 245
Table 6.70 Refactoring techniques based on each internal criterion by Team 1 .......... 247
Table 7.71 Refactoring techniques based on each internal criterion by Team 2 .......... 247
Table 6.72 Ranking the refactoring techniques by company A (internal attributes) .... 248
Table 6.73 Ranking the refactoring techniques by company B (internal attributes) .... 249
Table 6.74 Ranking the refactoring techniques by company C (internal attributes) .... 250
Table 6.75 Refactoring techniques based on each internal criterion by company A .... 252
Table 7.76 Refactoring techniques based on each internal criterion by company B .... 252
Table 6.77 Refactoring techniques based on each internal criterion by company C .... 252
Table 6.78 Ranking the refactoring techniques by Team 1 (external attributes) ....... 257
Table 6.79 Ranking the refactoring techniques by Team 2 (external attributes) ....... 258
Table 6.80 Refactoring techniques based on each external criterion by Team 1 ........ 260
Table 6.81 Refactoring techniques based on each external criterion by Team 2 ........ 260
Table 6.82 Ranking the refactoring techniques by company A (external attributes) .. 261
Table 6.83 Ranking the refactoring techniques by company B (external attributes) .. 262
Table 6.84 Ranking the refactoring techniques by company C (external attributes) .. 263
Table 6.85 Refactoring techniques based on each external criterion by company A ..... 265
Table 6.86 Refactoring techniques based on each external criterion by company B ..... 265
Table 6.87 Refactoring techniques based on each external criterion by company C ..... 265
Table 6.88 Examples of refactoring techniques ................................................. 266
Table 6.89 Number of refactoring was used for each iteration by Team 1 ............... 267
Table 6.90 Number of refactoring was used for each iteration by Team 2 ............... 267
Table 6.91 Refactoring impact on the internal attributes by Team 1 ....................... 268
Table 6.92 Refactoring impact on the internal attributes by Team 2 ....................... 268
Table 6.93 Refactoring impact on the external attributes by Team 1 ...................... 269
Table 6.94 Refactoring impact on the external attributes by Team 2 ...................... 269
Table 6.95 Number of refactoring was used for each iteration by company A ......... 270
Table 6.96 Number of refactoring was used for each iteration by company B .......... 271
Table 6.97 Refactoring impact on the internal attributes by company A ................. 271
Table 6.98 Refactoring impact on the internal attributes by company B ................. 271
Table 6.99 Refactoring impact on the external attributes by company A ............... 272
Table 6.100 Refactoring impact on the external attributes by company B .................. 272
Table 6.101 Refactoring impact on time for company A and B ............................... 273
Table 6.102 Level of automated testing by Team 1 .............................................. 282
Table 6.103 Level of automated testing by Team 2 .............................................. 283
Table 6.104 Automated testing levels by Team 1 based on each criterion individually .... 284
Table 6.105 Automated testing levels by Team 2 based on each criterion individually .... 284
Table 6.106 Level of automated testing by company A .......................................... 285
Table 6.107 Level of automated testing by company B .......................................... 286
Table 6.108 Level of automated testing by company C .......................................... 286
Table 6.109 Automated testing levels by company (A) based on each criterion .......... 288
Table 6.110 Automated testing levels by company (B) based on each criterion .......... 288
Table 6.111 Automated testing levels by company (C) based on each criterion .......... 288
Table 6.112 Release indicators ranking by Team 1 .................................................. 291
Table 6.113 Release indicators ranking by Team 2 .................................................. 292
Table 6.114 Release indicators by Team 1 based on each criterion individually ........ 294
Table 6.115 Release indicators by Team 2 based on each criterion individually ........ 294
Table 6.116 Release indicators ranking by company A .......................................... 295
Table 6.117 Release indicators ranking by company B .......................................... 296
Table 6.118 Release indicators ranking by company C .......................................... 296
Table 6.119 Release indicators by company (A) based on each criterion individually ... 298
Table 6.120 Release indicators by company (B) based on each criterion individually ... 298
Table 6.121 Release indicators by company (C) based on each criterion individually ... 298
Table 6.122 Acceptability level of AHP as a decision tool ...................................... 308
Table 6.123 Acceptability level of AHP ranked by Team 1 ...................................... 309
Table 6.124 Acceptability level of AHP ranked by Team 2 ...................................... 310
Table 6.125 Acceptability level of AHP ranked by the three companies .................. 311
Table 6.126 Linking the data to the research questions and propositions .................. 312
LIST OF FIGURES

Figure 1.1 Research Structure ................................................................. 8
Figure 2.2 XP Lifecycle ........................................................................... 15
Figure 2.3 AHP Structure....................................................................... 21
Figure 2.4 Screenshot of expert choice comparison tool............................ 30
Figure 3.5 Distribution of AHP themes ..................................................... 35
Figure 3.6 Distribution of AHP based on the application areas.................. 35
Figure 3.7 Distribution of AHP by fields.................................................... 38
Figure 3.8 Distribution of AHP by integrated methods............................... 38
Figure 3.9 Distribution of AHP among the integrated methods.................. 40
Figure 3.10 Distribution of AHP in software system areas based on the purpose of use .... 84
Figure 3.11 Comparing integrated AHP with stand-alone AHP.................... 84
Figure 4.12 Essential elements in the XP empirical study in education .......... 123
Figure 6.13 AHP Structure for prioritizing user stories in the first study........ 167
Figure 6.14 Prioritizing user stories from the developers and customer’s perspectives ..... 172
Figure 6.15 AHP structure for prioritizing user stories from the customer’s perspective ... 173
Figure 6.16 AHP structure for prioritizing user stories from the developer’s perspective ... 174
Figure 6.17 Importance of the criteria for the customers in the user stories prioritization... 176
Figure 6.18 Importance of the criteria for the developers in the user stories prioritization.. 176
Figure 6.19 AHP Structure for ranking the prioritization techniques................ 183
Figure 6.20 Importance of the criteria for Team 1 in the prioritization techniques ........ 185
Figure 6.21 Importance of the criteria for Team 2 in the prioritization techniques ........ 186
Figure 6.22 AHP structure for selecting design tool ..................................... 193
Figure 6.23 Importance of the criteria for Team 1 in the design tools selection ...... 194
Figure 6.24 Importance of the criteria for Team 2 in the design tools selection ...... 195
Figure 6.25 AHP structure for ranking the CRC card for Team 1 .................... 200
Figure 6.26 Importance of the criteria for Team 1 in the CRC cards ............. 201
Figure 6.27 AHP Structure for ranking the CRC cards for Team 2 .................. 205
Figure 6.28 Importance of the criteria for Team 2 in the CRC cards ............. 205
Figure 6.29 AHP Structure for selecting pairs ........................................... 213
Figure 6.30 Importance of the criteria for Team 1 in the pair selection ......... 215
Figure 6.31 Importance of the criteria for Team 2 in the pair selection ........... 216
Figure 6.32 Importance of the criteria for company A in the pair selection ...... 218
Figure 6.33 Importance of the criteria for company B in the pair selection .................................. 219
Figure 6.34 Importance of the criteria for company C in the pair selection .................................. 219
Figure 6.35 AHP structure for the rules in pair matching ............................................................... 223
Figure 6.36 Importance of the criteria for Team 1 in the rules of pairing ...................................... 225
Figure 6.37 Importance of the criteria for Team 2 in the rules of pairing ...................................... 226
Figure 6.38 Importance of the criteria for company A in the rules of pairing .................................. 228
Figure 6.39 Importance of the criteria for company B in the rules of pairing .................................. 229
Figure 6.40 Importance of the criteria for company C in the rules of pairing .................................. 229
Figure 6.41 AHP Structure for the refactoring techniques based on the internal attributes .......... 241
Figure 6.42 Importance of the internal criteria for the refactoring techniques by Team 1 ............. 244
Figure 6.43 Importance of the internal criteria for the refactoring techniques by Team 2 ............. 245
Figure 6.44 Importance of the internal criteria for the refactoring by company A ....................... 248
Figure 6.45 Importance of the internal criteria for the refactoring by company B ....................... 249
Figure 6.46 Importance of the internal criteria for the refactoring by company C ....................... 250
Figure 6.47 AHP Structure for the refactoring based on the external attributes ......................... 256
Figure 6.48 Importance of the external criteria for the refactoring by Team 1 .............................. 257
Figure 6.49 Importance of the external criteria for the refactoring by Team 2 .............................. 258
Figure 6.50 Importance of the external criteria for the refactoring by company A ...................... 261
Figure 6.51 Importance of the external criteria for the refactoring by company B ...................... 262
Figure 6.52 Importance of the external criteria for the refactoring by company C ...................... 263
Figure 6.53 AHP Structure for automated testing decision .......................................................... 280
Figure 6.54 Importance of the criteria for Team 1 in automated testing ....................................... 282
Figure 6.55 Importance of the criteria for Team 2 in automated testing ....................................... 283
Figure 6.56 Importance of the criteria for company A in automated testing .................................. 285
Figure 6.57 Importance of the criteria for company B in automated testing .................................. 286
Figure 6.58 Importance of the criteria for company C in automated testing .................................. 286
Figure 6.59 AHP structure for release indicators ranking ............................................................. 290
Figure 6.60 Importance of the criteria for Team 1 in release indicators ........................................ 291
Figure 6.61 Importance of the criteria for Team 2 in release indicators ........................................ 292
Figure 6.62 Importance of the criteria for company A in release indicators .................................. 295
Figure 6.63 Importance of the criteria for company B in release indicators .................................. 296
Figure 6.64 Importance of the criteria for company C in release indicators .................................. 296
**LIST OF ABBREVIATIONS**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AHP</td>
<td>The Analytic Hierarchy Process.</td>
</tr>
<tr>
<td>QFD</td>
<td>Quality Function Deployment.</td>
</tr>
<tr>
<td>LP</td>
<td>Linear Programming.</td>
</tr>
<tr>
<td>MILP</td>
<td>Mixed integer Programming.</td>
</tr>
<tr>
<td>GP</td>
<td>Goal Programming.</td>
</tr>
<tr>
<td>GA</td>
<td>Genetic Algorithms.</td>
</tr>
<tr>
<td>ANN</td>
<td>Artificial Neural Network.</td>
</tr>
<tr>
<td>DEA</td>
<td>Data Envelopment Analysis.</td>
</tr>
<tr>
<td>ANP</td>
<td>Analytic Network Process.</td>
</tr>
<tr>
<td>ACP</td>
<td>Airline Control Program.</td>
</tr>
<tr>
<td>IT</td>
<td>Information Technology.</td>
</tr>
<tr>
<td>TPF</td>
<td>Transaction Processing Facility.</td>
</tr>
<tr>
<td>COST</td>
<td>Commercial Off-The-Self.</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphic User Interface.</td>
</tr>
<tr>
<td>SPICE</td>
<td>Software Process Improvement and Capability dEtermination.</td>
</tr>
<tr>
<td>SAG</td>
<td>Security Activities Graph.</td>
</tr>
<tr>
<td>ISM</td>
<td>Interpretive Structure Modeling.</td>
</tr>
<tr>
<td>PCRs</td>
<td>Process Change Requests.</td>
</tr>
<tr>
<td>CMM</td>
<td>Capability Maturity Model.</td>
</tr>
<tr>
<td>CMMI</td>
<td>Capability Maturity Model Integration.</td>
</tr>
<tr>
<td>GAHP</td>
<td>Group AHP.</td>
</tr>
<tr>
<td>ROI</td>
<td>Return On Investment.</td>
</tr>
<tr>
<td>SDLC</td>
<td>Software development life cycle.</td>
</tr>
<tr>
<td>ARCH</td>
<td>Architecture.</td>
</tr>
<tr>
<td>EVNT</td>
<td>Event Notification.</td>
</tr>
<tr>
<td>AUTH</td>
<td>Authentication.</td>
</tr>
<tr>
<td>SECU</td>
<td>Security.</td>
</tr>
<tr>
<td>ERP</td>
<td>Enterprise Resource Planning.</td>
</tr>
<tr>
<td>KM</td>
<td>Knowledge Management.</td>
</tr>
<tr>
<td>QAT</td>
<td>Quality Attributes Technique.</td>
</tr>
<tr>
<td>PQR</td>
<td>Product Quality Risk.</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>MAS</td>
<td>Multimedia Authorizing Systems.</td>
</tr>
<tr>
<td>BPR</td>
<td>Business Process Reengineering.</td>
</tr>
<tr>
<td>IS</td>
<td>Information System.</td>
</tr>
<tr>
<td>FMS</td>
<td>Flexible Manufacturing System.</td>
</tr>
<tr>
<td>ATAM</td>
<td>Architecture Tradeoff Analysis Method.</td>
</tr>
<tr>
<td>PSNT</td>
<td>Public Switched Telephone Network.</td>
</tr>
<tr>
<td>RFQ</td>
<td>Request For Quotation.</td>
</tr>
<tr>
<td>RDE</td>
<td>Technology Development Envelope.</td>
</tr>
<tr>
<td>FDDI</td>
<td>Fiber Distributed Data Interface.</td>
</tr>
<tr>
<td>DQDB</td>
<td>Distributed Queue Dual Bus.</td>
</tr>
<tr>
<td>HAZOP</td>
<td>Hazard and Operability Studies.</td>
</tr>
<tr>
<td>FMEA</td>
<td>Failure Mode and Effects Analysis.</td>
</tr>
<tr>
<td>FTA</td>
<td>Fault Tree Analysis.</td>
</tr>
<tr>
<td>ETA</td>
<td>Event Tree Analysis.</td>
</tr>
<tr>
<td>MCDA</td>
<td>Multiple-Criteria Decision Analysis.</td>
</tr>
<tr>
<td>TR</td>
<td>Traditional Requirement.</td>
</tr>
<tr>
<td>IIBA</td>
<td>International Institute of Business Analyst.</td>
</tr>
<tr>
<td>CRC</td>
<td>Class Responsibility Collaboration.</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modeling Language.</td>
</tr>
<tr>
<td>BDUF</td>
<td>Big Design Up Front.</td>
</tr>
<tr>
<td>RPD</td>
<td>Role-Play Diagram.</td>
</tr>
<tr>
<td>BLR</td>
<td>Binary Logistic Regression.</td>
</tr>
<tr>
<td>LOCC</td>
<td>Lines of Code for Class.</td>
</tr>
<tr>
<td>NOF</td>
<td>Number of Fields.</td>
</tr>
<tr>
<td>NOM</td>
<td>Number of Methods.</td>
</tr>
<tr>
<td>LOCM</td>
<td>Lack of Cohesion in Methods.</td>
</tr>
<tr>
<td>DIT</td>
<td>Depth of Inheritance Tree.</td>
</tr>
<tr>
<td>NOTC</td>
<td>Number of Test Cases.</td>
</tr>
<tr>
<td>TDD</td>
<td>Test-Driven Development.</td>
</tr>
</tbody>
</table>
CHAPTER 1
INTRODUCTION

This chapter introduces the overall research questions and structure of this thesis. First, the research problem is discussed in section 1.1. Second, the research motivation is briefly described in section 1.2. Third, the research objectives are presented in section 1.3. Fourth, the research method is presented in section 1.4. Fifth, the thesis’ original contributions are presented in section 1.5. Finally, the last section, 1.6, includes the structure of this thesis, which might be useful to refer to when reading it.

1.1 Research Problem

As software development becomes more and more complex, the need for more sophisticated collective decision making tools that facilitate the development process is more urgent than ever [1]. This is especially true for agile methods like Extreme Programming (XP), for which many methods have been devised to extend its scope to larger and/or more complex projects. Drury et al. [2] conducted a study with 43 agile developers and managers to analyze decisions made during the iteration cycle and to identify obstacles to these decisions. They found that conflicting priorities was one of six key obstacles to these decisions.

Improving XP requires improving each of the practices comprising it. Looking at each practice individually and investigating the factors which affect this practice, including the decision criteria, will enable us to discover where the conflicts come from and how Analytical Hierarchy Process (AHP) can improve the decisions involved.
There are five main XP processes which involve various activities such as selection, ranking, evaluation and decision. These practices are: planning game, pair programming, simple design, refactoring, and testing.

Until recently, the majority of the decisions have been made based on voting or experience. “The downside of using voting as a decision making mechanism is that someone (or some group of someone) will lose [sic]. The win/lose mechanism can create a divide within the team which will pull it apart over time or generate passive aggressive behavior” [3]. Tessem [4] researched the influence of experience in the decision-making process by interviewing four different types of the software development organizations in Norway. The results confirmed that experience has a significant influence on developers’ empowerment, and showed that the experts in these organizations participated indecision making more than other developers. Additionally, the author emphasized a very important concept: “Agile methodologies are supposed to increase developers’ say in not only the design and solution but also in how project are organized, what tasks to priorities, and not least, the selection of work tasks in the daily work”[4]. Moreover, Drury and Orla [5] conducted 34 interviews and 18 observations across four agile project teams in three countries - India, Sweden and Ireland - to investigate how agile project teams make decisions. They found that the agile teams did not use a rational, linear process when making decisions. As stated by Drury, “Often, people’s experience drives their individual and group decisions and agile teams do not identify and evaluate a series of options, as the linear process outlines” [5].

A more mature method is needed due to the fact that quick responses to changes may create problems among team members when making decisions. “Teams can struggle
with creating a consensus, become passionately split on an idea or simply struggle to understand the options that are appropriate. Each team needs a pallet of decision making tools available (or a coach with those techniques) or risk failing to make good decisions or decisions in a timely manner” [3]. Mocz [6] attempted to answer the question, “Why isn’t agile working?” He discussed three flaws of the agile methodology. One of these flaws is responding to change over following a plan. He stated, “In theory, developers code while collaborating with stakeholders to define, refine and change requirements as the projects goes along. The methodology, however, does not distinguish between big and small changes. Every change has a cost, but agile does not account for this. The result? People often change really big things late in the game using the rationale that since it’s an agile project, it can handle it. The only way the project can handle this is by adding iterations. As that happens, defects that might have been easy to fix at one point get harder and harder to fix, since the code base keeps changing” [6]. Similarly, ignoring team members’ opinions may be another reason for failure [7][8]. James and Shane stated, “Almost every challenge in building great software is, in some way, a people problem. That challenge may be communicating effectively, dealing with the unpredictability of moods and motives, or figuring out how to harness people’s desire to do the right thing for the team and the project. Few problems are solely technical [...] within the team, anyone can be a leader. Encourage team members to turn to the person or people most qualified to make a necessary decision” [9]. Different levels of decisions, different decision makers, and different types of decisions are forcing problem diagnosis to be studied by each practice individually, which will eventually improve XP as a process of software development. Coyle et al. [10] pointed out that “despite the many
benefits associated with group work, groups are often subjected to process losses such as groupthink, which in turn have a negative impact on group decision making. Limited prior research has suggested that such process losses may be even more prevalent in agile software development characterized by highly cohesive, self-managing teams” [10].

The research problems addressed in this thesis can be succinctly described as follows:

P1. Weakness in identifying the tangible and intangible criteria or factors affecting the decisions in XP practices.
P2. Difficulty in deciding which alternative should be chosen when different options are involved.
P3. Lack of structure in the decision problems encountered by XP teams, including the developers, managers, and customers.
P4. Relying only on the voice of experts in software development neglects XP values such as courage and communication.
P5. Lack of existing guidance to help XP teams aggregate several evaluations by different people who have various roles in the development.
P6. Lack of a formal justification of the ranking or prioritizing of items in XP practices.
P7. Lack of a framework based on mathematical aspects that serve different activities for all these five practices.

Given these points, the AHP method was introduced to XP to help both customers and developers make effective decisions.
1.2 Motivation

Extreme Programming (XP) is a very popular method for software development [9]. XP offers a set of practices designed to work together and provide tangible value to the customer. It brings all team members including managers, customers, and developers, together to create a very flexible and collaborative team. XP stands on five essential pillars: communication, simplicity, feedback, respect, and courage. XP claims to give software developers the ability to deliver a product in incremental steps while maintaining quality at a very high level [9]. XP’s ‘extreme’ quality results from taking 12 well-known practices to their logical extremes [11]. In this context, the Analytic Hierarchy Process (AHP) can be used to support the XP values and facilitate the achievement of XP’s goals. AHP has already been adopted in many software system applications for various purposes such as decision assessment and measurement, prioritization, selection, and evaluation [12,13,14,15,16], but it seldom has been used in agile processes. This thesis explains how to apply AHP to XP on a practice-by-practice basis.

Some of the advantages of AHP are shown below.

- AHP can structure a complex decision in XP practice that involves many factors and criteria.
- AHP enables us to identify tangible and intangible criteria such as objective criteria, subjective criteria, risk information, preferences, technical data, benefits, and cost estimates.
- AHP can measure the interactions among the criteria affecting XP practice in a simple way.
• AHP combines judgments to obtain the relative importance of alternatives.
• AHP can aggregate a group of judgments to come up with the final results.

1.3 Research Objectives

The main objective of this research is to introduce an AHP-based methodology into Extreme Programming to support making complex decisions. This methodology can be applied in various activities during the XP development process. Our aim is to prevent random decisions and increase the cohesiveness of the XP team. Put simply, AHP ensures that everybody’s voice in an XP team be heard.

Our goal is to highlight the factors which affect XP practices when a decision is made, and evaluate these factors using a formal method. In addition, we elucidate the alternatives that decision makers face and provide them with a way of selecting, ranking, and evaluating these alternatives. The intent of this research is to define and establish a road map to use AHP in lightweight software processes like XP. By investigating the use of AHP in XP practices, we hope to show that it is possible to approach decision making in XP in a more cohesive and coherent way through a judicious mix of objectives organized into a formal hierarchy.

The research described in this thesis aims to determine whether an XP practice requires structured decision-making and to propose AHP as the method to fulfill this requirement. This need can appear in the form of selection, ranking, evaluation or decision choices. Our solution will encourage everyone in the team, including the developers and customers, to participate in the decisions related to development. The use of AHP in decision making ensures the consideration of all the factors influencing a
decision.

1.4 Methodology and Scope

The majority of this research is based on case studies. The case study is one of the most important research methodologies for software engineering [17][18]. Layman stated that “case studies are valuable because they involve factors that staged experiments generally do not exhibit, such as scale, complexity, unpredictability, and dynamism” [19]. To accomplish the goal of this work, the case study was the research method of choice. However, before we can draw any conclusions about the use of AHP in the development of XP, we must review multiple case studies to provide sufficient results. Two case studies in the academic environment as well as three in an industrial setting make up the case study analysis. Figure 1.1 illustrates the how these case studies cooperated.

Some of the XP practices are descriptive concepts, such as: code standard, 40-hours of work, and continuous integration. These practices are excluded from the study. This research mainly focuses on the core of the XP: planning game, pair programming, simple design, refactoring and testing. In some of the practices, AHP is used multiple times for different purposes. To investigate the potential use of AHP in XP projects, fully implemented software is developed in an educational environment at the graduate level. In addition, feedback from experts in the industry is obtained. The team size considered in the study was small or medium due to the fact that a large size requires an exponentially larger number of comparisons which complicates calculations.
Figure 1.1 Research Structure

* for applying the AHP results, we have chosen the refactoring practice to be validated in the industry.
1.5 Original Contributions

This thesis is expected to provide XP teams with a framework based on the Analytic Hierarchy Process that can be injected into several XP practices. This framework will provide different solutions for various problems that are encountered during software development, including decisions, ranking and selections.

Relying only on expert managers or senior developers can create a big gap in understanding among team members. AHP will allow everyone’s voice to be heard. Introducing AHP into XP development is expected to assist in capturing the subjective and objective factors that affect specific XP practices. The potential contributions of this research are as follows:

1. Contextualizing the criteria and alternatives that affect XP practice decisions.
2. Establishing an AHP-based framework in order to structure XP practice decision-making problems.
3. Introducing AHP to 10 areas related to XP practices.
4. Determining the decision and ranking areas in the core XP practices (planning game, simple design, pair programming, refactoring, testing) and providing the following:
   4.1 AHP prioritization model for implementing user stories.
   4.2 AHP selection model for the simple design tools.
   4.3 AHP prioritization model for the CRC cards.
   4.4 AHP decision model for applying the pair matching roles.
   4.5 AHP selection model for choosing the best pairs.
   4.6 AHP evaluation model for the refactoring methods (two models).
4.7 AHP evaluation model for releasing indicators.

4.8 AHP decision model for automated testing level.

5. Investigating the advantages and disadvantages of using AHP with XP.

6. Practicing the XP process in an educational environment with a real industrial project and real customer involvement.

1.6 Organization of Thesis

This thesis is organized into seven chapters. Chapter 1 provides an introduction and motivation of the problem. It also presents the objectives of the research, the methodology and the scope, and summary of the original contributions.

Chapter 2 provides fundamental background information about the agile methodology, and defines extreme programming and its value and practices. Chapter 2 also introduces AHP and the Expert Choice tool.

Chapter 3 presents the suitability of AHP to software development process. It also reviews the uses of AHP in various fields, including the IT and software fields. This chapter highlights the benefits of AHP brought specifically to software development as well.

Chapter 4 discusses the applicability of the XP method in educational environments, focusing especially on post-secondary schools. Several empirical studies were investigated to explore the possibility of making the educational environment in line with XP principles and values.

Chapter 5 describes the set up of the educational and industrial studies used in this thesis. It covers the participants’ background, duration of the studies, the development process, communication tools, and customer roles. This chapter also includes the
information about the project that was assigned for the XP development.

Chapter 6 provides the proposed areas in the XP practices which may benefit from using AHP. It shows planning game, simple design, pair programming, refactoring, and testing as the main potential areas for applying AHP for various purposes. Chapter 6 provides the XP-AHP case studies’ results and participants’ feedback.

Chapter 7 draws some final conclusions about the research in this thesis. This chapter also discusses possible future areas for research exploration as related to this work.
CHAPTER 2
LITERATURE REVIEW

This chapter provides fundamental background information on XP and AHP. It begins with the basic history of the agile methodology and its value. It then provides a definition of XP and its value and practices. The second part of the chapter introduces the Analytic Hierarchy Process (AHP), the method that will be applied to various areas in XP in the thesis. It also shows the AHP structure, the pairwise comparison process and other necessary information related to this method.

2.1 Agile Methodology

In 2001, at The Lodge at Snowbird ski resort in the Wasatch mountains of Utah, a group of 17 experienced and recognized software development “gurus” [20] met and discussed lightweight development methods. They then defined the approach now well known as Agile Software Development. After that, they published the Manifesto for Agile Software Development [21, 22].

2.1.1 Existing agile methods

Agile software development includes the following methods [23]:

1. Agile Modeling.
3. Extreme Programming (XP).
7. Scrum.
8. Lean Software Development.
10. Adaptive Software Development.

Each of these agile methods has its own set of principles, values, and practices. However, XP is the most well defined method that provides a clear approach for developers to do their daily work. XP is more advanced than the other agile methods in terms of frequency of use, and demonstrated success. By 2003, 66% of the software projects around the world were using agile methods and 90% of those projects were using XP [24][25].

2.2 Extreme Programming

Extreme Programming is one of the most successful methods in software development. XP offers a set of practices designed to work together to provide a tangible value to the customer. It brings all team members including the managers, customers, and developers into one strong node to create a very informative and collaborative team. XP stands on five essential pillars: communication, simplicity, feedback, respect, and courage. Software developers who use XP can deliver a product confidently faster along while maintaining very high product quality [20]. XP has received substantial attention in the industry and proved its efficiency and effectiveness in small and mid size projects.

2.2.1 Extreme programming process

The life cycle of XP consists of five phases [23]:

1. Exploration: Customers and developers sit together to explore the desired features in the system. The customers write stories as system requirements and the
developers ask questions to ensure that the developed system fulfills the customer’s needs. The technical environment and other tools are prepared in this phase.

2. Planning Phase: Customers start prioritizing the user stories to be implemented based on their business value. When stakeholders agree on the features of the system, the stories are then broken down into smaller tasks. The developers estimate the time required to develop each user story. Then they commit to tasks and continue working side by side with the other XP team members.

3. Iterations to Release: The iterations are defined. The schedule for each iteration along with the stories to be developed is established. The customers decide in which iteration they’d like to release the stories. The plan for the development and the technical process are created.

4. Productionizing: The core development activities (design, coding, refactoring, and testing) occur in this phase. Changes are welcome and customers can add features or remove them from the system.

5. Maintenance: Customers receive support and new people may be added to the team, thus changing the team structure.

6. Death Phase: This is the last phase, when the customers are satisfied with the systems and features being developed, i.e., when they don’t have any more changes for the product. However, death may also occur when the team cannot fulfill the customer’s needs and the system cannot be developed.
2.2.2 Extreme programming values

Extreme Programming recognizes five values, as detailed in “Extreme Programming Explained” [26] as follows:

1. Simplicity

According to Kent Beck “simplicity” is the most important value in XP: solve today’s problem with a simple solution; use a simple design to code only for today’s needs. The “YAGNI” (you are not going to need it) approach is used. Simplicity can improve team communications and welcome changes during the entire development process.

2. Communication

Building successful software requires effective communications among the development team. In XP, the developers and customers must always be on the same
page and share the same views about the system. All team members must obtain the information at the right time.

3. Feedback

Feedback is quite related to simplicity and communications. Feedback can be accomplished by the system through testing activity. By having the unit tested, the system can provide the developers with the needed information regarding the system’s state. The customer can also give feedback through the acceptance test written with the user stories. Also, feedback from the whole team is important when customers add or change some of the system requirements.

4. Courage

Courage is injected into most XP practices. When developers decide to apply the simple design concept, they need courage to do so. When they refactor their code, they must have enough courage to take such a decision. People in XP work together, but individual inputs are recognized. When two people work in a pair on coding, they need to be free and fair when making mistakes or asking questions.

5. Respect

XP team members should have self-respect and should respect the work and efforts of others. Everybody should be appreciated and know that his/her voice is valuable. However, no one should make changes that break the system without collective decision. Loyalty to the team is always recommended, and motivation from managers and organization is necessary.
2.2.3 Extreme programming practices

In this section, XP practices are briefly presented [26,27]. The following descriptions are the starting points for the application of AHP.

1. Planning game

The customers and developers cooperate to produce maximum value as rapidly as possible. The communication between the developers and customers is formalized. The customers write the system’s requirements as user stories. The developers can help the customers to explain exactly what the system needs. The developers initially estimate these stories using story points to identify the complexity and cost for implementation. The user stories are also broken down into smaller tasks. Then the customers prioritize these stories based on value in order to identify which user stories should be implemented first.

2. Simple design

Simple design holds that developers should design only for today and make it simple as possible to get the job done. Unnecessary and complex code should be removed immediately. Simple design has the following four characteristics:

- It runs all the tests.
- It has no duplicate code.
- It expresses all the programmers’ ideas very clearly.
- It contains a smaller number of classes and methods.

3. Small Release

Produce a simple version of the system that has useful features. Release early and often. Every integration generates a release and keeps adding a new feature to extend the
system gradually.

4. **System metaphor**

The team should come up with a metaphor that can be understood by people who have no prior technical knowledge using very common names and vocabulary. This metaphor should help to clarify how the system works in a easy way.

5. **Pair programming**

Two programmers work together on one workstation. Whoever writes the code is named the “driver” and the partner is named the “navigator”. The navigator watches and reviews the code written by the driver. The roles switch frequently between the two programmers.

6. **Test-driven development**

Before the programmers start coding they first write the test. When the test is run the job is done. The most important tests in XP are unit tests and acceptance tests. Unit tests are written by the developers and performed using automated tools. The acceptance tests (known as functional tests) are written by the customers and test the entire system to make sure the features being developed meet the customer’s needs and desires. The user story cannot be considered as a complete until it passes the acceptance test. Acceptance tests should also be automated.

7. **Refactoring**

Refactoring involves changing how the code looks, not how it works. It involves altering the internal structure and keeping the external functionality without changes. This can be done by removing duplication and simplifying the readability.
8. **Collective ownership**

No one can claim that he/she owns the code. Any developer can work on any part of the code anytime.

9. **Continuous integration**

The system is being constantly integrated and built all the time. All changes made in the codebase are integrated every day. The test must run completely before and after the integration.

10. **40-Hour work week**

Overtime work is not recommended. The developers should go home on time. However, it is acceptable to work overtime once in a while. If it occurs very often, it does mean that the team has a problem needs to be fixed and the overtime will not be the solution.

11. **Coding standards**

Rules of coding are emphasized and programming languages are chosen collectively. The team should apply these rules throughout the project. The developers apply the same format and style into the source code to achieve consistency during development.

12. **On-site costumer**

The customer in XP plays a central role in the software development process. His involvement should be in full and at all times available. If the team doesn’t have this type of customer, they assign one of them to be a proxy customer.
2.3 Analytic Hierarchy Process (AHP)

AHP is a systematic approach which represents a problem involving the consideration of multiple criteria in a hierarchical model with different levels. AHP reflects human thinking by grouping the elements of the problem when confronted with a complex decision. AHP is a theory of measurement and decision-making developed by Thomas L. Saaty. The value of AHP lies in its ability to combine or synthesize quantitative as well as qualitative considerations in an overall evaluation of alternatives. As such, it can be especially applicable to the evaluation of complex system designs involving software, hardware, and humans that can easily include hundreds of system quality indicators [28]. AHP is very beneficial for decision makers since it helps capture both subjective and objective evaluation measures; this provides a useful mechanism for checking the consistency of evaluation measures and alternatives, and reduces bias in decision-making [29]. The real value of AHP lies in its ability to combine or synthesize quantitative as well as qualitative considerations in the overall evaluation of alternatives. As such, it can be especially applicable to the evaluation of complex system designs involving software, hardware, and humanware that can easily include hundreds of system quality indicators. AHP has emerged as a powerful technique for determining relative worth and ranking among a set of elements [30].

AHP comprises the following steps:

1. Structure the hierarchy model for the problem by breaking it down into a hierarchy of interrelated decision elements; see figure 3.3.

2. Define the criteria or factors and construct a pairwise comparison matrix for them; each criterion on the same level is compared with other criteria in respect to their
importance to the main goal.

3. Construct a pairwise comparison matrix for alternatives with respect to each objective in separate matrices.

4. Check the consistency of the judgment errors by calculating the consistency ratio.

5. Calculate the weighted average rating for each decision alternative and choose the one with the highest score.

2.3.1 Pairwise comparison

The pairwise comparison approach was proposed by Saaty (1980) and has long attracted the interest of many researchers in various areas and applications. Pairwise comparisons are mainly used to determine the relative importance of each alternative in the context of given criteria [29]. Table 2.1 shows the numerical scale developed by Saaty:
Table 2.1 AHP Numerical scale developed by Saaty [29]

<table>
<thead>
<tr>
<th>Scale</th>
<th>Numerical Rating</th>
<th>Reciprocal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equal importance</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Moderate importance of one over</td>
<td>3</td>
<td>1/3</td>
</tr>
<tr>
<td>other</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Strong or essential importance</td>
<td>5</td>
<td>1/5</td>
</tr>
<tr>
<td>Very strong or demonstrated</td>
<td>7</td>
<td>1/7</td>
</tr>
<tr>
<td>importance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extreme importance</td>
<td>9</td>
<td>1/9</td>
</tr>
<tr>
<td>Intermediate values</td>
<td>2,4,6,8</td>
<td>1/2, 1/4, 1/6, 1/8</td>
</tr>
</tbody>
</table>

The following can be noticed from the Saaty scale:

1. The value of 9 as the upper limit and 1 as the lower limit; “Psychological experiments have also shown that individuals cannot simultaneously compare more than seven objects (plus or minus two) (Miller, 1956)” [29].

2. The AHP comparison scale commonly uses the values of (1,3,5,7,9) due to the difficulty in distinguishing between two very close values of importance. However, using the values 2,4,6, and 8 when needed is not wrong at all.

If we have three objects to be compared (X, Z, Y), we will have three comparisons, if it is more than three it will follow Table 2.2:
An AHP Matrix is a matrix whose rows and columns have the same parameters. If \((c_1)\) is in the first row then \((c_1)\) also in the first column, and if the \((c_2)\) in the second row then \((c_2)\) will also be in the second column, etc., as shown in Table 2.3.

### Table 2.3 AHP criteria in matrix

<table>
<thead>
<tr>
<th></th>
<th>(c_1)</th>
<th>(c_2)</th>
<th>(c_n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(c_1)</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(c_2)</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>(c_n)</td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

#### 2.3.2 Priorities derivation

All the AHP criteria are compared in a comparison matrix, and the decision alternatives are compared in a comparison matrix with respect to each criterion. These comparisons give the local weights of the criteria and alternatives. The global weights of
the decision alternatives are calculated by the product sums of the local attribute weights and alternative weights.

The following explains how to make the reciprocal matrix from pairwise comparison [31,32]:

\[ A = \begin{pmatrix}
    \frac{w_1}{w_1} & \frac{w_1}{w_2} & \cdots & \frac{w_1}{w_n} \\
    \frac{w_2}{w_1} & \frac{w_2}{w_2} & \cdots & \frac{w_2}{w_n} \\
    \vdots & \vdots & \ddots & \vdots \\
    \frac{w_n}{w_1} & \frac{w_n}{w_2} & \cdots & \frac{w_n}{w_n}
\end{pmatrix} \]

The upper triangle matrix will be filled as:

\[ A = \begin{pmatrix}
    \frac{w_1}{w_1} & \frac{w_1}{w_2} & \cdots & \frac{w_1}{w_n} \\
    \frac{w_2}{w_2} & \frac{w_2}{w_2} & \cdots & \frac{w_2}{w_n} \\
    \vdots & \vdots & \ddots & \vdots \\
    \frac{w_n}{w_n} & \frac{w_n}{w_n} & \cdots & \frac{w_n}{w_n}
\end{pmatrix} \]

The matrix \( A = (a_{ij}) \), \( a_{ij} = \frac{w_i}{w_j} \), \( i,j = 1, \ldots, n \), where \( n \) is a positive number.

The diagonal elements of the matrix \( A \) always score 1 as follows:
So, when \( a_{ij} \) is an element of a row \( i \) and column \( j \), then the lower diagonal can be completed using the following formula [31]:

\[
a_{ji} = \frac{1}{a_{ij}}
\]

Note that all the comparison matrix elements must be \( a_{ij} \geq 0 \).

When the comparison matrix is filled, we compute the priority vector, the normalized Eigenvector of the AHP matrix.

2.3.3 Consistency in AHP

As comparisons make sense only if derived from consistent or near consistent matrices, a consistency check must be applied to indicate the level of coherence between judgments. When assigning values to the AHP matrix, whether to compare the criteria or the alternatives, decision makers should be consistent in their rating of preferences. In this step, we should measure the consistency of subjective judgment. For example, A, B and C are compared as follows [33]:
Logically, if \( A > B, \ B < C \), then \( A \) must be preferable to \( C \). This logic is called the “transitive property” [33]. If the comparison is transitive, it means the judgment will mostly likely be consistent. On the contrary, if \( C > A \), it means the judgment is not consistent.

Comparison matrix \( A \) can be consistent when [31]:

\[
a_{ij} \times a_{jk} = a_{ik}
\]

for all values of \( i, j \) and \( k \).

However, if this is the case, we should not force the decision maker to make a judgment, i.e. if it is not transitive. There are several pieces of AHP software in the market which apply consistency, forcing the concept to be more rational. Examples include the MakeitRational software, which asks the decision makers to adjust their value if it is not transitive.

For matrices involving human judgments, AHP assumes that evaluators don’t know the actual weights, represented by vector \( w \). “So the condition \( a_{ij} \times a_{jk} = a_{ik} \) doesn’t hold as human judgments are inconsistent to a greater or lesser degree” [34].
A vector $w$ of order $n$ will be $Aw = \lambda w$. For such a matrix, $w$ is an eigenvector (of order $n$) and $\lambda$ is an eigenvalue. The matrix is consistent when $\lambda = n$. The $w$ vector appears in the equation as $A \cdot w = \lambda_{\text{max}} w$, and $\lambda_{\text{max}} \geq n$, where $A$ is the pairwise comparison matrix and $\lambda_{\text{max}}$ is the largest eigenvalue of matrix $A$. The difference, if any, between $\lambda_{\text{max}}$ and $n$ will make the judgment inconsistent. On the other hand, if $\lambda_{\text{max}} = n$, then that means the judgments are consistent [34].

Saaty stated that for the consistent reciprocal matrix, the largest value is equal to the size of the comparison matrix, or $\lambda_{\text{max}} = n$. For measuring consistency, Saaty proposed a consistency index (CI), which is related to the eigenvalue method as follows [35]:

$$CI = \frac{\lambda_{\text{max}} - n}{n - 1}$$

The consistency ratio, the ratio of CI and RI, is given by [35]:

$$CR = \frac{CI}{RI}$$
If the CR <= 0.10, the degree of the consistency in the pairwise comparison matrix is acceptable and satisfied, but if the CR>0.10, the results are not meaningful.

Saaty [35] calculated the RIs as shown in Table 2.4.

<table>
<thead>
<tr>
<th>n</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>RI</td>
<td>0.58</td>
<td>0.9</td>
<td>1.12</td>
<td>1.24</td>
<td>1.32</td>
<td>1.41</td>
<td>1.45</td>
<td>1.49</td>
</tr>
</tbody>
</table>

### 2.3.4 Advantages of using AHP

The following presents some benefits AHP can bring to the practice [29,36]:

1. Structuring a complex problem that involves many factors and criteria to facilitate the process of making a decision and to simplify the relations among the structure levels.

2. Ability to identify and evaluate tangible and intangible criteria such as objective criteria, subjective criteria, risk information, preferences, technical data, benefits, and cost.

3. Measuring the interactions among the criteria using the pairwise comparison approach.

4. Combining the judgments to obtain the relative importance of the alternatives.

5. Ability to aggregate the group of judgments to come up with average results.
2.3.5 Expert Choice tool

Expert Choice was the core calculation tool used in this work to obtain the results. The following is a brief description about the tool extracted from the manual:

“Expert Choice software is designed to help decision-makers overcome the limits of the human mind to synthesize qualitative and quantitative inputs from multiple stakeholders. The result is sound analysis and clear thinking that everyone in the organization agrees with and understands.

The general Expert Choice software features: [sic]

- Brings structure and measurement to your decision-making processes
- Helps you determine strategic priorities and ensure decisions are tightly aligned.
- Communicates priorities and builds consensus around decisions
- Documents and justifies your decisions
- Enables you to move forward quickly and confidently

With Expert Choice software and services, structure and repeatable processes can be implemented to enable organizations to collaborate around decisions to increase alignment with strategic objectives, buy-in around decisions, and the ability to move forward quickly with confidence” [37].

However, for this thesis the web-based software version named “Comparion™ Core” was chosen rather than the desktop application. The best feature about the online application is that it enables the user to collaborate with his/her team in real-time to prioritize objectives, evaluate alternatives, and incorporate quantitative data. Comparison Suite is based on the .NET framework from Microsoft and supports Windows 2003 servers.
A print screen of the application is captured can be seen in Figure 2.4.

![Figure 2.4 Screenshot of expert choice comparison tool](image)

2.3.6 Expert Choice main functions

As can be seen in Figure 2.4, there are six main functions utilized by the Expert Choice software: structure, measure, synthesize, allocate, iterate, and report. Each function provides various features for AHP users. The developers have provided in detail a manual and tutorial for each function to make it clear for advanced users and beginners as well. The information presented in this section can be found in the Expert Choice manual available on its website [37].

- **Structure**

  Expert Choice implements AHP to make the best decisions in reaching a goal. Expert Choice enables decision makers to structure the problem complexity, to measure
the importance of the criteria, and to weigh the alternative preferences with respect to these criteria and main goal. The first step in making a choice of the most preferred alternative is structuring the problem and allocating resources to a combination of alternatives. This step helps to analyze the factors that affect the decision. Structuring a problem involves identifying criteria, separating alternative courses of action into a hierarchy, determining the contribution of each alternative to each criterion, and identifying the people participating in AHP evaluation as well as their roles.

- **Measurement**

  After structuring the hierarchy of criteria and alternatives, priorities are obtained for relative preference of the criteria as well as the relative importance of the alternatives with respect to the criteria. Generally, measurements in Expert Choice are performed by making pairwise relative comparisons to derive the priorities of the objectives in the same level, taken two elements at a time. The software enables both subjective as well as objective measurement. This feature makes Expert Choice a somewhat unique software due to the fact that most mathematical models don’t allow for human judgment to the extent possible with AHP and Expert Choice. Moreover, all the measurements obtained by Expert Choice are ‘ratio level measures’ to avoid calculations that lead to mathematically meaningless results.

- **Synthesis (Results)**

  A synthesis (combining) of the measures is done by Expert Choice automatically according to the objectives hierarchy following the structuring and measurement steps. This synthesis includes both objective information based on available hard data and subjective input based on the knowledge, experience, and judgment of the participants.
The synthesis results are organized to present priorities for the competing objectives and overall priorities for the alternatives. Because of Expert Choice’s structuring and measurement methods, the results using this tool are mathematically sound, unlike results using traditional approaches like spreadsheets.

- **Allocate**

  If someone decides on one course of action, this feature can be skipped. However, it is a good option for those who want to combine some actions. It is a powerful feature to be able to determine the optimal combination of actions or investments subject to constraints such as budget, space, materials, personnel and dependencies.

- **Iterate**

  The decisions that Expert Choice deals with are complex, important and sometimes require iteration. This feature is perfect and powerful for this purpose. For example, when someone feels that alternative Y is much preferable to alternative X because of a specific factor, and this factor was not included as an objective, it can be added to the model easily. Then changes can then be applied to the structure automatically.

- **Report**

  Through this feature, Expert Choice will generate a full report for the project, including the project information and final results.
CHAPTER 3

SUITABILITY OF AHP TO SOFTWARE DEVELOPMENT PROCESSES

This chapter contains an analysis of the suitability of AHP for software development processes. The analysis is based on more than 400 papers discussing the applications of AHP in many disciplines, including manufacturing, management, industry, government and engineering. In particular, 49 papers directly relate to software development. This chapter presents three extensive literature reviews on AHP methodology published in 2006, 2008, and 2010. These articles present the use of AHP across different themes and in different application areas.

3.1 First Study

Vaidya and Kumar [38] reviewed 150 applications of AHP published in highly reputed international journals since 2003. In this article, the authors classified the AHP applications into three groups as follow: (1) AHP applications based on themes such as selection, decision making, evaluation, priority and ranking, allocation, planning and development, and cost-benefits analysis; (2) specific AHP applications such as medicine and forecasting; and (3) AHP applications integrated with other methodologies like Quality Function Deployment (QFD). Tables 3.5 and 3.6 show the number of AHP articles classified by their themes and applications areas.
3.1.1 Summary and charts for the first study

Table 3.5 AHP articles based on their themes

<table>
<thead>
<tr>
<th>Purpose of using AHP (Themes)</th>
<th>Number of Articles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selection</td>
<td>32</td>
</tr>
<tr>
<td>Evaluation</td>
<td>26</td>
</tr>
<tr>
<td>Planning and Development</td>
<td>18</td>
</tr>
<tr>
<td>Benefit- Cost</td>
<td>8</td>
</tr>
<tr>
<td>Priority and Ranking</td>
<td>20</td>
</tr>
<tr>
<td>Quality function Development</td>
<td>7</td>
</tr>
<tr>
<td>Decision</td>
<td>20</td>
</tr>
<tr>
<td>Medicine</td>
<td>5</td>
</tr>
<tr>
<td>Allocation</td>
<td>10</td>
</tr>
<tr>
<td>Forecasting</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 3.6 AHP articles based on their application area

<table>
<thead>
<tr>
<th>AHP Application Areas</th>
<th>Number of Articles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Personal</td>
<td>26</td>
</tr>
<tr>
<td>Social</td>
<td>23</td>
</tr>
<tr>
<td>Manufacturing</td>
<td>18</td>
</tr>
<tr>
<td>Political</td>
<td>6</td>
</tr>
<tr>
<td>Engineering</td>
<td>26</td>
</tr>
<tr>
<td>Education</td>
<td>11</td>
</tr>
<tr>
<td>Industry</td>
<td>15</td>
</tr>
<tr>
<td>Government</td>
<td>13</td>
</tr>
<tr>
<td>Others</td>
<td>12</td>
</tr>
</tbody>
</table>
Figure 3.5 shows how the AHP methodology is distributed by theme.

![Figure 3.5 Distribution of AHP themes](image)

Figure 3.6 shows how the AHP methodology is distributed among by application area.

![Figure 3.6 Distribution of AHP based on the application areas](image)
It is observed from the first study that AHP is being used primarily in the theme areas of selection, evaluation, priority and ranking, and decision. In terms of the application areas, it seems AHP has been used more in the engineering, personal and social categories. The use of AHP in decision-making areas has been rapidly increasing. Some governments like India and China are using the AHP applications for developing their own countries, where evaluation and complex economic decisions are needed. This is evidence that AHP is a concept which decisions makers can be rely on [38].

3.2 Second Study

William Ho [39] reviewed AHP applications integrated with other tools published from 1997 to 2006. Five decision tools commonly combined with AHP were deeply analyzed in the article. These were: (1) mathematical programming including linear programming (LP), integer linear programming (ILP), mixed integer programming (MILP), and goal programming (GP); (2) quality function deployment (QFD); (3) meta-heuristics including genetic algorithms (GA) and artificial neural network (ANN); (4) SWOT analysis; and (5) data envelopment analysis (DEA) [39]. The following tables 3.7 and 3.8, show the number of integrated AHP articles reviewed by the authors separated by field.
3.2.1 Summary and charts for the second study

Table 3.7 AHP publications by field

<table>
<thead>
<tr>
<th>AHP Fields</th>
<th>Number of Articles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logistics</td>
<td>21</td>
</tr>
<tr>
<td>Manufacturing</td>
<td>18</td>
</tr>
<tr>
<td>Government</td>
<td>4</td>
</tr>
<tr>
<td>High education</td>
<td>4</td>
</tr>
<tr>
<td>Business</td>
<td>3</td>
</tr>
<tr>
<td>Environment</td>
<td>3</td>
</tr>
<tr>
<td>Militarily</td>
<td>3</td>
</tr>
<tr>
<td>Agriculture</td>
<td>2</td>
</tr>
<tr>
<td>Health-care</td>
<td>2</td>
</tr>
<tr>
<td>Marketing</td>
<td>2</td>
</tr>
<tr>
<td>Industry</td>
<td>1</td>
</tr>
<tr>
<td>Service</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3.8 AHP publications based on integrated methods

<table>
<thead>
<tr>
<th>Integrated methods</th>
<th>Number of Articles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mathematical Programming</td>
<td>33</td>
</tr>
<tr>
<td>QFD</td>
<td>16</td>
</tr>
<tr>
<td>Meta historic</td>
<td>8</td>
</tr>
<tr>
<td>SWOT</td>
<td>5</td>
</tr>
<tr>
<td>DEA</td>
<td>4</td>
</tr>
<tr>
<td>Total</td>
<td>66</td>
</tr>
</tbody>
</table>
Figure 3.7 shows the AHP methodology distribution based on fields.

Figure 3.7 Distribution of AHP by field.

Figure 3.8 shows how the AHP methodology is distributed by integrated method.

Figure 3.8 Distribution of AHP by integrated method.
From the second study, it was found that integrated AHPs have been successfully applied to a wide variety of fields and applications. Due to its simplicity and great flexibility, AHP can be combined with other techniques, such as mathematical programming, QFD, meta-heuristics, SWOT, and DEA. The authors observed that logistics and manufacturing are the two main areas of AHPs’ integration.

3.3 Third Study

Sipahi and Timor [40] reviewed 232 applications selected from more than 600 related papers published in the period 2005-2009. This study covered all AHP and analytic network processes (ANP) and their extensions, such as fuzzy AHP and fuzzy ANP. The authors classified the articles by country, publication year, application area and integrated tools. Since we are only analyzing the AHP methods and their applications, Table 3.9 shows the number of the AHP applications reviewed by the authors along with their integrated methods.
3.3.1 Summary and charts for the third study.

<table>
<thead>
<tr>
<th>Integrated Method</th>
<th>Number of Articles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation</td>
<td>14</td>
</tr>
<tr>
<td>TOPSIS</td>
<td>7</td>
</tr>
<tr>
<td>GIS</td>
<td>10</td>
</tr>
<tr>
<td>Goal Programming</td>
<td>5</td>
</tr>
<tr>
<td>DEA</td>
<td>12</td>
</tr>
<tr>
<td>Delphi Method</td>
<td>4</td>
</tr>
<tr>
<td>Balanced Scorecard</td>
<td>4</td>
</tr>
<tr>
<td>Factor Analysis</td>
<td>4</td>
</tr>
<tr>
<td>Fuzzy logic Model</td>
<td>5</td>
</tr>
<tr>
<td>Genetic Algorithm</td>
<td>5</td>
</tr>
<tr>
<td>SWOT</td>
<td>3</td>
</tr>
</tbody>
</table>

Figure 3.9 shows how the AHP methodology is distributed among the integrated methods.

![Figure 3.9 Distribution of AHP among the integrated methods.](image-url)
The authors found that the use of the AHP technique has continued to increase exponentially. AHP was integrated with Simulation, DEA, and GIS methods more than any other methods presented in the articles. Overall, the authors proved that AHP is a reliable and powerful technique which has been utilized by managers and other users to make better decisions. The researchers are predicted that the use of AHP as a decision support tool will keep increasing worldwide.

3.4 AHP in Software System Applications

In previous studies, authors have proved that AHP is a reliable and powerful technique used by managers and other users to make better decisions. However, it has been noted that AHP in software development has not been studied deeply compared to other fields. Therefore, this section focuses precisely on AHP applications in software systems or applications somehow related to the area of software development. The applications were categorized based on their purposes of use. Most of the AHP applications in software system fall in one of the following categories:

1. Decision
2. Assessment and measurement
3. Prioritization
4. Selection
5. Evaluation
6. Other
3.4.1 Decision

Roper-Lowe and Sharp [41] used AHP to choose between computer operations systems in the field of information technology decision. The method was precisely applied on a British Airways system that deals with 1.2 million cargo shipments every year. The airline had a computer system called BA80 to manage the cargo operations. The system was 10 years old and used the IBM Airline control program (ACP) operation system. After consulting the IT and management experts, British Airways decided to move on to another more advanced system. Naturally, discussion regarding advantages and disadvantages of whether the decision would be worth it or not was brought to the table. Three options were available to the managers, out of which they had to choose one. The first option was to not change the system and keep the old one (ACP), coping its shortcomings. The second option was to take advantage of the Transaction Processing Facility TRF2 system, which has eight processors and uses the SNA communications protocol. The third option was to take a risk and try a high technology system by IBM called DB2. AHP was introduced in this case to help in choosing the best system, taking into consideration the benefits and the cost hierarchy. Eight experts participated in structuring the AHP and chose five main decision criteria in the model as follows: ease of conversion, performance, flexibility, speed of development, and risk. Rather than through questionnaires, interviews were conducted for weighting the criteria and scoring the elements.

At the end of the study, the authors determined that the AHP was a very helpful tool for understanding the problem and revealing some issues which had not been explicitly made clear in the initial analysis. However, the authors also found it difficult to
compare tangible attributes with intangible ones. Additionally, the study clearly showed that AHP might hide some important interdependencies, since it assumed that the all attributes on the same level were independent.

Karlström and Runeso [42] introduced AHP to Extreme Programming decision in an organization. This study showed the possibility for using for XP practices in a real-world environment. The case study was conducted in 2001 in a software engineering course at Lund University. The participants of this study were two different groups of people: twenty-six students and five developers from an industry. There were two possible ways for applying the XP practices in the development. The first way was to introduce all of the XP practices at once. The second way was to select only some of the XP practices based on their suitability to the environment. The authors proved that the AHP method supported managers in making such a decision and helped them analyze the output data obtained from the AHP framework.

The hierarchy here was structured based on two main factors: (1) effectiveness and (2) easiness. The two groups (students and developers) rated the XP practices with respect to these two categories. The authors presented the XP practices in detail to the subject of the study. Then, the participants rated the reflection of each practice to the main factors and evaluated the alternatives for each situation. The final strategic decision was made based on the results of the AHP analysis.

From this study, the authors found that students and developers have different points of view which affect the evaluation process. The key conclusion however, was that AHP analysis results in a better visualization of opinions, which helps tremendously in making decisions. This study had only one main purpose: to decide which XP practice the
organization should implement. It did not, however, discuss any subsequent decisions encountered after deciding on which XP practice to implement. The present research extends the use of AHP to further decisions were missed.

Ahmad and Laplante [43] introduced AHP to select appropriate software project management tools. The criteria and factors affecting such a decision were structured and analyzed to identify the measurements of several aspects in software development.

The main goal was stated at the top of the hierarchy, which was to select the best tools for software project. Twelve factors were specified in the AHP structure as follows: (1) task scheduling; (2) resource management; (3) collaboration; (4) time tracking; (5) estimating; (6) risk assessment; (7) change management; (8) reporting chart; (9) file attachment; (10) e-mail notification; (11) process/methodology; and (12) portfolio management. These factors were selected based on the most common features presented by the most commercial off-the-shelf solution (COTS).

The alternatives were not specified clearly; thus, the authors assumed T1, T2, T3, T4, and T5 to be the alternatives to complete the AHP model and calculate the results from the comparison matrix tables. Some experts participated in prioritizing the value of each factor. For facilitating the problem analysis and evaluating the qualitative data, the authors used an MS Excel macro instead of using advanced tools like Expert Choice. The study ended up providing the managers with a reliable framework for evaluating the appropriate software project management tools.

Information technology management has become a complex task for many companies, and IT resources are considered as main success factor for business firms (Neilson, 1990) [44]. As a result, a systematic analysis for evaluating IT components is
needed for the sake of cost effective information systems. Udo [44] proposed AHP as a tool for analyzing the outsourcing decisions of information systems.

He assumed that IT outsourcing decisions are based on five criteria: (1) strategic importance, (2) stakeholder’s interest, (3) vendor issues, (4) cost operations, and (5) industry environment. The three alternatives in the AHP structure were: (1) outsourcer, (2) IT department, and (3) end user. At the top of the AHP hierarchy was ‘selecting the IT management technique’. Additionally, the model helped to analyze the effects of the IT service on achieving the strategic goal.

Expert Choice was used for the mathematical calculation and related graphics. Five experts were asked to evaluate the three alternatives based on the given criteria. Moreover, a sensitivity analysis of AHP was performed to display a real-time interactive graph. Expert Choice created five different graphical modes: (1) performance, (2) dynamic, (3) gradient, (4) two-dimensional plot, and (5) differences.

At the end of this analysis, comprehensive comparisons led to the conclusion that allowing the IT department to manage itself was the most recommended. On the other hand, the end users were ranked the lowest among the three options presented in the study. The authors concluded that AHP is a suitable method for analyzing information technology outsourcing decisions.

Hwang et al. [45] used AHP as a part of developing web-based multi-criteria to analyze make-or-buy decision problems for a manufacturing system. A GUI-type simulation software was built to drive decision alternatives from reviewers using the brainstorming method.

The web-based development worked in two main steps: (1) determining the rank-
ordered priority for make-or-buy decisions using the AHP model; and (2) integrating special problems with several objectives, criteria, and attributes using the fuzzy approach. There were six criteria affecting the make-or-buy decision presented by the authors, obtained from a separate literature review: (1) cost, (2) quality, (3) delivery speed, (4) delivery reliability, (5) volume flexibility, and (6) product flexibility.

Along with the AHP approach, the fuzzy method was used to weigh each decision factor and prioritize the alternatives. AHP was a very helpful tool for the decision makers in that it allowed them to maximize the benefits gained from the online-based decision support system and overcome this particular problem.

3.4.2 Assessment measurement

Yu and Heng [46] used AHP in order to determine an organization’s current level of agility and how much agility is needed. AHP helped in structuring the problem of analyzing the agility factors affecting decisions, along with observing the experts’ experience and knowledge. A method called BBN was used alongside AHP to evaluate the probabilities of reaching a high agility score.

The authors established an index system for agility measurement which included all the decision factors and attributes. Experts participated in giving values to the factor weights and translating the quantitative statement into numbers. The hierarchy of the agility measurement built in four main factors: time, cost, robustness, and scope of change. Each factor has sub-factors as follows: (A) Time (time from: (1) market opportunity to discovering it; (2) discovering the opportunity to completion of the product; (3) the completion of the product to the market; (4) the new product hitting the
market to payment); (B) Cost (cost of: (1) information, (2) research, (3) production, (4) selling, and (5) and other); (C) Robustness (Quality robustness, profit robustness, completion rate for the order); and (D) Scope of change (adaptability for: (1) the information system, (2) the production, (3) the cooperation, (4) and the employee). After following the steps of AHP and mathematical calculation, the authors found that AHP is an effective method and a rational tool that provides results for measuring the agility of a manufacturing enterprise. These factors and elements guided the researchers to assess the agility of the enterprise.

Lee and Kim [47] used AHP to assess system performance and obtain a unified index. The AHP model was structured with three states: (1) ideal, (2) actual, and (3) possible states. The unified index was obtained by calculating the eigenvalue for the three states.

Beside the benefit of evaluating the system performance, the author found that AHP can be applied to the large distributed system and help in the planning phase.

Jun [48] proposed AHP as a systemic approach that can identify the boundaries between the adjacent ratings (largely, partially, not achieved) in SPICE (Software Process Improvement and Capability dEtermination) assessments. “Based on analysis of SPICE phase 2 trials data, El Emam and Jung (2000) reported that the difficulties expressed by competent assessors were 24.56%, 43.96% and 21.05% in understanding a boundary between Fully and Largely, between Largely and Partially, and between Partially and not achieved, respectively” [48]. To overcome these difficulties, Jun suggested to re-investigate evidence and collect more information. So, he reviewed the assessment model (ISO/IEC 15504:Part5) used in the SPICE projects, which is structured in two
dimensions: process and capability. The data was collected from sixteen assessors in two SPICE-based assessments in Korea. Lists of questionnaires were given to the participants and the responses were received and organized. The assessors encountered three boundary problems, two between the Largely and Partially and the third one between Fully and Largely. The AHP structure was able to reduce the difficulties in distinguishing between these two adjacent ratings scales.

The most valuable finding brought by the author in his study is that AHP is an approach which can determine the assessors’ consensus in boundary problem and reduce the assessment effort.

Moreover, Thalia et al. [48] used AHP to develop the most essential security matrices in order to measure the security of an information system. A new algorithm was created using the hierarchy structure model, and was based on the decomposition methodology discussed in detail in [48]. AHP helps in deciding the relative importance of all the sub-goals. The algorithm benefited from the quantitative evaluation model with qualitative decisions to identify the security level in the information system field.

3.4.3 Prioritization

Byers and Shahmehr [49] developed a framework based on AHP to prioritize the security activities in the software development process and help the developers improve their software lifecycle. AHP was also used to select the optimal set of security activities to meet the organization’s goals. Other security concepts discussed in the paper are extracted from the AHP analysis, such as vulnerabilities and sustainable software security processes (S3P). The cost and fitness factors were taken into consideration for the AHP
structure. Other important criteria were discussed for the purpose of selecting appropriate activities such as sufficiency and optimality. During the interview of the S3P users, the authors found that the costs assigned to the activities varied among the users. They also noticed that the alternatives were difficult to measure directly. The authors established a structure for the problem of selecting the best way to prevent vulnerability, and called it Security Activities Graph (SAG). From the SAG, the “cost assignment” was presented as a problem needing to be structured using the AHP hierarchy. The authors formulated three levels to understand this problem. The top level was the goal (cost assignment). The second level was the criteria (fit, expense, and time). The third level was the alternatives (use strncpy, use strlcpy, use s-string, use bstring, and don’t copy). The researchers took AHP further to integrate it with a scatter search instance implemented in the SGA simulation. AHP was found to be a very helpful tool when assigning costs to the activities in a set of SAGs and analyzing the criteria affecting cost.

In another article, Sadiq et al. [50] used the AHP approach for the purpose of eliciting the software requirements and its prioritization. Extracting requirements from the user can be a hassle and a reason for failure. In order to apply the AHP approach, the authors gathered information from the user and prioritized the system requirements. In this study there were eight criteria evaluated before selecting an elicitation method. These criteria are as follows: (1) adaptability, (2) computer-aided software engineering, (3) stakeholders acceptance, (4) easy implementation, (5) quick implementation, (6) shallow learning curve, (7) high maturity, and (8) scalability.

The authors mentioned more than ten software requirement techniques such as misuse cases, quality function development, controlled requirements expression, issue-
based information systems (IBIS), and others. Only three were used for eliciting the requirements in the case study.

AHP was found to be the most effective technique in two areas: (1) high performance and (2) ease. These two factors are very attractive features of AHP when eliciting appropriate software requirements. Properly eliciting system requirements and prioritizing them according to the framework used in this study was able to reduce system errors [50].

In order to prioritize the requirements and identify the important elements of software application, Perini et al. [51] performed an empirical study to compare the accuracy of AHP and another method named CBRanking (Case-Based Ranking).

Eighteen students participated in this study as follows: 12 PhD students at the University of Trento and 6 Junior Researchers at IRST (a public research center in Italy). The object of the study was a web application called CoCoA, which delivers a personalization service for audio compilation. The number of requirements was 20. Each was given a simple textual description such as: “save a user defined compilation” and “multilingual interface” [51].

The authors defined accurate prioritization as when a person produces a priority order that reflects the decision maker’s opinion. They analyzed the properties of each technique and measured the accuracy for the requirements prioritization, asking how close the resulting ranking was to their opinions. The results showed that the AHP method was more accurate than CBRanking. Among the 18 participants, 28% stated that CBRanking was more accurate than AHP, 0% found them equally accurate, and 72% stated that AHP was more accurate.
In a project by the Korean government, Lee et al. [52] used AHP techniques and Data Envelopment Analysis (DEA) to prioritize the relative weights of energy technology for the purpose of evaluating energy efficiency. AHP was used to structure such a complex problem and evaluate all the quantitative and qualitative factors in energy efficiency.

On the other hand, DEA as an evaluation and assessment tool was used to integrated several input and output data. The goal of the AHP model was to prioritize energy technologies of energy efficiency plan. In the AHP hierarchy, three levels were presented. In level 1, five factors were presented as follows: (1) UNFCCC, (2) economic spin-off, (3) technical spin-off, (4) urgency technology development, (5) and quantity of energy use. In level 2, the sub-factors were presented: (1) possibility of developing technology, (2) potential quantity of energy savings, (3) market size, (4) ease of technology spread. In level 3, five alternatives were named as follows: Tech1, Tech2, Tech3, Tech4, and Tech5. The two combinations of the AHP model and DEA technique provided a comprehensive analysis which led to a proper prioritization of the energy technology among many alternatives. The results definitely satisfied the decision makers in terms of preference ratings.

In another article, Kanungo and Monga [53] suggested that the AHP-based model be used to rate projects and processes in an organization of high maturity (level 4 and level 5) in CMM and CMMI. Precisely, they used AHP (GAHP) and Interpretive Structure Modeling (ISM) to prioritize the importance in process change requests (PCRs) in a software process improvement program. The continuous improvement in a mature process relies on the PCR system with the PCR framework as the main component.
The AHP model enables organizations to perform relative ranking of their projects and processes, while the ISM methodology helps in identifying the relationship among ideas.

Four criteria were introduced to evaluate the PRCs: (1) reducing effort; (2) improving quality of process deliverables; (3) innovating practices for different types of projects; and (4) providing inputs on streamlining processes to make them consistent and structured. Also, there were six alternatives of PCR, as listed in the following:

1. Provide scenario-based configuration management guidelines for different category of projects.
2. SRS process must also include use-cases way of capturing and managing the functional requirements of the software system.
3. Refine the guideline for selection of review types to achieve optimal balance.
4. Guidelines and criteria for preparing a plan for development of the project plan should be developed.
5. Guidelines for selection of appropriate levels of testing with associated benefits should be addressed in the testing process.
6. To review project plan template, which would improve delivery controls and a ‘final inspection’ that ensures that all required inspections/tests have been completed.

This study was done at IBM Global Service, Bangalore, India, since IBM is level 5 in the CMM and CMMI. The GAHP method was proposed to rank the PCRs in IBM and select the best one. Different experts participated in the evaluation and established the pairwise tables. However, due to the GAHP techniques used in this study, the pairwise comparisons were reduced. Additionally, the technique didn’t take into the account the degree to which judgment A was stated to be better than judgment B.
Eventually, the AHP concept provided a robust framework for the organizations to help them overcome the obstacles raised by the practitioners. The consensus was also easy obtained in the priority structure. The authors presented many relative aggregation frameworks and outlined several weaknesses.

Vargas [54] stated that organizations face difficulty when attempting to come up with the right translation for the phrases “low cost” and “high benefits”. Low cost can be mean anything which is cheaper, less complex, or with less risk, and high benefits could be mean more profits, greater return on investment or increased market share. As a result, this makes the decision an extremely challenging step in science and technology (Triantaphyllou, 2002). In Vargas’ study, AHP was proposed to prioritize and select the projects in a portfolio. Decisions are generally based on values and preferences, so the authors suggested that the AHP method be used to help the organization decide which project should be executed. The authors chose the development of a fictitious decision model for the ACME organization to construct the AHP hierarchy and prioritize projects.

A set of criteria was presented with the assumption of applying the method in four areas: finance, strategy, planning, and project management. This was decided upon to determine the appropriate meaning of benefits and low cost. The following 12 criteria were selected and grouped into four categories:(1) financial, which includes Return on Investment (ROI), profits, (USD), and Net Present Value; (2) strategic, which includes the improved ability to compete in international markets, improved internal processes, and improved reputation; (3) stakeholder commitment, which includes team commitment, organizational commitment, and project manager commitment; and (4) other criteria, which includes urgency, low risks, and internal technical knowledge. The relative
importance of the criteria and relative weights to the global goal were calculated.

The author applied the AHP structure to the case where the ACME organization had six different projects to be prioritized. These projects were: (1) moving to a new office facility; (2) implementing a new ERP system; (3) opening an office in China; (4) developing a new product; (5) IT infrastructure outsourcing; and (6) running a new local marketing campaign. When all the AHP processes were run, it was decided that the highest priority for the ACME Organization based on the evaluation of all given projects was to move to a new office [54].

In this study, the authors showed how AHP could benefit the project managers during the analysis phase by giving them a clear vision and understanding of the complexity involved in project decisions. They also found that AHP helped to predicate decisions about a portfolio in negotiation, human aspects and strategic analysis.

3.4.4 Selection

Software reliability metrics is an essential concept in field of software evaluation and assessment. Li et al. [55] used AHP for modeling the problem of how to select appropriate software reliability metrics in each phase of software development. Five selection criteria were proposed as follows: (1) relevance (to reliability), (2) experience, (3) correctness, (4) practicality, and (5) feasibility. These criteria played a main role in the four phases of the development process: requirement specification phase, design phase, implementation phase, and testing phase.

Five experts provided grading for thirty-one software reliability matrices based on the five criteria mentioned previously. The authors indicated the difficulties faced by the
experts when grading the degree of each metric. However, they finally achieved synthetic weights of each matrix. The relative importance was constructed via a 1-9 scale and used comparison matrices for the criteria. In each phase, the top-ranked metrics were identified and analyzed.

The consistency and sensitivity of the comparison matrixes in the AHP method helped the authors to decide on which metrics were appropriate for software reliability. Results showed that “Fault Density”, “Test Coverage”, and “Error Indices” were ranked as the top metrics. The authors praised AHP as a tool for selecting appropriate metrics correctly, stably and systemically [55].

Yang and Lin [56] used AHP to select the best partner for a virtual enterprise using a multi-agent system. The authors identified a partner selection system that included two sections: optimizing and negotiating. The optimizing section was more important than the negotiating section, and thus AHP was used as core tool in selecting the best partner [56].

Additionally, Babu et al. [57] looked at the software architecture evaluation process as a phase of the decision-making approach. They took advantage of the AHP model to investigate several issues in software architecture, such as the evaluation iterative process, quality of attributes, flexibility, changing requirements, and available technology for implementation. The idea was to meet the functional and nonfunctional requirements and select an architectural style for a given software system. The AHP model in this research was integrated with global programming and resulted in a mathematical model that optimized the selection of software architecture. They focused on the architecture quality criteria by conducting an assessment for several phases of the software development life cycle (SDLC).
Some specialists (stakeholders) ranked and gave priority weights for the whole design alternatives, and their justification was considered an important activity. As a result, they positively enhanced the selection process and improved the system quality. By combining global programming and the AHP model, the authors developed an excellent model for evaluating the architecture design over a set of alternatives.

Five majors factors were taken into the consideration: Architecture (ARCH), Event Notification (EVNT), Authentication (AUTH), Security (SECU) and HETR. Besides that, the levels of the AHP-GP structure were examined and compared from two perspectives: cost and technology. The authors found the model very useful in reaching their goal.

Management Information Systems are very important in business administration as they have a direct impact on the organization’s profits [58]. Therefore, implementing a Management Information System essentially requires a stable and reliable approach which investigates the different factors that play a role in the efficiency of the business. Hu and Xia [58] applied AHP to research in this area. The idea was to place the problem and main factors into a hierarchical model in order to come up with a decision. The AHP structure in this study had three levels: goal, factors, and alternatives. The goal - choosing the MIS implementation approach - was placed at the top of the AHP structure. There were four factors contributing to the goal and four alternatives achieving the decision; these were placed in the second and third levels, respectively. The four factors were: (1) performance, (2) quality, (3) costs, and (4) benefits. Each factor was broken down into sub-factors. The performance factor was divided into (1) usability, (2) efficiency, and (3) stability. The quality factor included (1) flexibility, (2) maintainability, and (3)
scalability. The cost factor included (1) fee and (2) time. The benefits factor included (1) training, (2) reducing, and (3) optimization. The AHP model comprised four alternatives: (1) user development, (2) joint development, (3) outsourcing development, and (4) purchasing software packages [58].

The factors, sub-factors, and alternatives were compared to each other to determine their relative importance. The consistency of judgments was measured by the consistency ratio (C.R). After analyzing the data and the calculation process, the authors found that the purchasing software package ranked as the highest priority alternative for the best MIS implementation approach, with a priority of 0.331. The second rank was joint development with a priority of 0.308.

The Enterprise Resource Planning (ERP) system is an advanced piece of information technology which has a direct impact on the production and service method of enterprise. Selecting the appropriate ERP software has become a difficult task since any mistakes in the choice will cost a lot money and time [59]. Several activities besides the quantitative and qualitative attributes used in the ERP must be handled to support decisions on investment. Karaarslan and Gundogar [59] found AHP to be a reliable method that can help in selecting the best ERP software to meet the given requirements. The new ERP evaluation approach proposed in this study was used to establish ERP modules for production instead of using basic software criteria. This required experts to be involved in the evaluation. A case study was presented of a marble machine factory to validate the method. Around 45 offices, 86 production employees, and a turnover of $US 4 billion annually were targeted in the study. The company wanted to choose one of two ERP software packages to monitor the business process. The authors prepared
questionnaires given to the employees to recognize their expectations of the ERP modules. The managers from each department participated in the evaluation process.

Several modules were presented in this study: (1) financial modules, (2) production modules, (3) material management modules, (4) distribution modules, and (5) recourse modules. Moreover, eight main criteria were taken into the consideration: (1) general system specifications, (2) a production module, (3) material management module, (4) financial management module, (5) quality management module, (6) sales and distribution module, (7) maintenance management module, and (8) human resource module. The firm analysis and mathematical part is detailed in [59]. In the conclusion, the authors provided a framework to confidently choose the right ERP software package to improve the business process.

Lee and Choi [60] focused on selecting significant attributes and factors to examine some aspects in B2C electronic commerce. Precisely, the authors chose the AHP model to evaluate domestic fashion websites in Korea aiming to increase product quality and minimize costs through online business. The study used some of the traditional marketing mix elements to show the benefits of using the AHP structure and analyze the related sites issues.

Four criteria were presented and are discussed here: (1) product, (2) price, (3) place, and (4) promotion. Each factor was combined with decision alternatives as follows:

Product: simulation and brand loyalty.

Price: low price, electronic payment, and loss leader.

Place: transaction security, search and order delivery, and store management.
Promotion: interaction, publicity activity, and customer service.

The calculation of each factor was performed along with relative weights of the decision alternatives obtained. Additionally, pairwise comparison and preferences were carried out. The results showed that the low price and electronic payment alternatives were preferable compared to the others factors. The authors found that AHP was a powerful and robust tool that can help decision makers in electronic commerce gain a competitive edge in terms of quality and cost.

The AHP framework was also proposed for an organization to select the most appropriate Knowledge Management (KM) tools to support the innovation process [61]. The software market has a large number of KM tools which need to be analyzed and compared to find the best one, all while considering the technology, information and communication in KM. Grimaldi and Rippa [61] built an AHP structure that included three different organizational goals: (1) knowledge improvement, (2) performance improvement, and (3) network improvement. Four sub-criteria expressed the meaning of the criteria: (1) cost constraints, (2) time constraints, (3) functionality, and (4) reliability constraints. The alternatives were four sets of KM tools: business intelligence, content applications, data management tools, and collaborative tools. The AHP framework was applied to a medium enterprise located in Naples (Italy). This company always worked to respond quickly to the market change and requirements. The company wanted to adopt one of the KM tools to support the process of developing a new product. The AHP proposed by the authors sufficiently helped the Italian company to evaluate several KM tools and select one of them. The data was collected by interviewing the entrepreneur of the company, and was followed by the calculation of the weights and the relative
importance. The business intelligence was ranked (0.559) as the best KM tool, dominating the other alternatives for the criteria knowledge and performance. Data management tools were ranked second (0.213).

Chiam et al. [62] focused on evaluating and investigating the Quality Attributes Technique (QAT) and control product quality risk (PQR). They proposed a method based on AHP and risk management theory to achieve the goal of helping the software development team to choose the proper QAT for any quality domain across the life cycle.

This method investigated the QATs from three perspectives: (1) risk management, (2) process integration, and (3) cost/benefits. AHP was applied to evaluate the QAT alternatives from the perspective of cost/benefits. Four criteria were obtained from a previous study on the “quality attribute techniques framework” [63]. These criteria were: “(1) cost of application, including time required, cost of tools, documentation, test data cost; (2) expertise: (knowledge, experience, training); (3) number of performers, which includes individual or team, team size; and (4) impact:(failures detected, qualitative or quantitative results, execution time, ease of installation, modifiability, scalability, repeatability, test case completeness, test case precision)” [62]. The authors used the AHP model to evaluate four PQR assessment techniques: Hazard and Operability Studies (HAZOP), Failure Mode and Effects Analysis (FMEA), Fault Tree Analysis (FTA) and Event Tree Analysis (ETA). These four techniques were used as the alternatives in the AHP structure. The authors then applied the method to the safety-critical development project. The authors did not present the final results, but they claimed that AHP was a feasible method and that the results suited the QATs selected by experts in the field of engineering.
In the field of multi-media computing and technology, Lai et al. [64] used AHP to rank and prioritize three multi-media authorizing systems (MAS). AHP was applied in a computer service company that provides solutions for customers and the development team. Six experienced and trained engineers from the company’s multi-media unit participated in this study. Comprehensive questionnaires and interviews were given to the engineers to obtain the AHP formulation data. The participants came up with relevant criteria and sub-criteria affecting the decision in MAS evaluation. The AHP framework was structured into four levels. The highest level contained the main goal of selecting the best MAS from the lower levels. There were two main factors: (1) technical consideration and (2) managerial consideration; these were followed by four sub-factors for technical consideration: (1) development, (2) graphic support, (3) multi-media support, and (4) data file support. Factors for managerial consideration included: (1) cost effectiveness and (2) vendor support. Moreover, the three MAS alternatives were presented as product A, product B, and product C.

When the AHP framework was clearly established, the engineers were asked to perform pairwise comparisons in order to reach the best alternative mathematically. The relative weights and importance was calculated using Expert Choice. In the end, as the authors indicated, “The participants found that utilizing the AHP decision hierarchy to structure information and goals is a useful aid to understanding decision preferences and for driving discussion about them” [46]. The authors conducted t-test analyses and found that AHP was better than Delphi in six different areas: decision process contention, problem clarification, information utilization, goal elicitation, task comprehensiveness, and result contention. The participants were satisfied with the AHP, especially in the area
of decision quality (mean= 5.84) and user satisfaction (mean=5.89). However, AHP was not strongly supported in cost savings (mean=4.46).

Kim and Moon [65] created a functions model based on the AHP methodology to select a workflow management system for business process reengineering (BPR). This model aimed to help managers respond flexibly to the dynamic business environment. The main goal of the AHP structure was to select the best workflow management system for the firm. On the lower level, two sub-goals were presented: (1) control and (2) empowerment. The hierarchy structure clarifies the conflict between the control and empowerment; it also helps managers decide on which aspect will work better for their organization. The control aspect included three functions affecting the business process: (1) design (workflow entity, workflow rules, and design tool), (2) execution (static, dynamic), and (3) monitor (runtime, end-time). On the other hand, empowerment included four functions: (1) data (database, document), (2) people (synchronous, asynchronous), (3) application (EUC, program), and (4) machine (platform, peripherals). The authors discussed only the evaluation criteria of workflow design functions. First, the workflow entity had three functions: (1) document, (2) task, and (3) resource. Second, the workflow rules had three functions: (1) individual, (2) department, and (3) role. Finally, the design tool had three functions: (1) form, (2) script, and (3) GUI. There were more than 20 criteria associated with workflow design evaluation. The authors used a 37-question questionnaire survey to set the relative importance of all the functions. More than 300 experts participated in the survey gathered at a BPR conference.

AHP was used to analyze the criteria for selecting the best workflow for the organization’s business process reengineering project. The authors found that the
workflow functions for empowering individual employees (53.76%) was more important than those functions for controlling (46.24%).

Dorado et al. [66] aimed to solve the problem of software selection in an educational environment. Various software packages for computer simulations are used in student engineering labs. Thus, the author utilized the AHP structure to help select the appropriate software packages accordingly to educational criteria. The authors presented three main criteria associated with sub-criteria affecting the main goal: (1) software cost, including license price; (2) learning educational usability, including application, interface, availability, and portability; and (3) effectiveness, including self-correction, tutoring, and visualization.

The author applied this method to a subject for Thermal Engineering in University of Jaen. Alternatives were defined here as (S1, S2, S3, S4). The authors had years of experience in Thermal Engineering; this helped in collecting and analyzing the data. AHP for the software selection was structured via Multi-Criteria Decision Analysis. Pairwise comparison and other mathematical calculation were performed. The resulting selection was a software system named S1, which is designed to teach thermodynamic cycles.

The decision of choosing a single database from a huge selection is a very complicated decision for both end users and information specialists [67]. Tons of databases can be studied, but Zahir and Chang [67] studied only a number of online business database and hosts available in Canada. The authors designed and developed an expert system called ONLINE EXPERT and used AHP in order to prioritize the database and hosts for preference.

Searching for information is a key first step in any decision based on search and
selecting a suitable database. The expert system had five components: “(1) user interface, (2) interface engine, (3) explain module, (4) knowledge base: (a. knowledge base to user needs, b. enhanced knowledge base for selecting database matching users need), and (5) information base output generation” [67]. The authors used the AHP structure in the component of enhancement of the knowledge base. The authors grouped the database and selected the decision attributes for each group. The AHP structure was broken down into three levels. The decision goal was to select the best securities database. In the second level, there were three criteria: (1) number of stock exchanges covered, (2) when coverage begins, and (3) update frequency. Four alternatives were presented: (1) US stock market, (2) financial post securities info, and (3) the North American stock exchange. Similarly, AHP was used to determine which host was more favorable than others in terms of the number of the available databases and the range of information covered by these databases. So, two main criteria were defined: (1) number of databases and (2) number of database types. For the relative weights and pairwise comparison, the authors conducted interviews with four information specialists in Halifax. They also adjusted the weights using their method and collected some data from printed documents. The results were as follows: in terms of database name, both “The New York Times” and “Index of Legal Periodicals” were ranked the highest with a weight of 57 and the same host name: mead. “Canada Budgets Ontario Budgets” was ranked second with a weight of 52 and a host name of QL.

Schniederjans and Wilson [68] employed AHP as a framework to improve the process of selecting among information systems projects. Beside AHP, the Zero-One Goal Programming model (ZOGP) was used to consider some constraints that are not
really considered by AHP in the decision process, such as budget and resources. The ZOGP is another method which can help make an optimal selection. The goal was to prioritize six of information system (IS) projects. The AHP model was structured around the main objective of selecting the most desirable IS project while considering budget limitations. Four criteria were analyzed with respect to the main goal: (1) accuracy, (2) information processing efficiency, (3) promotion of organizational learning, and (4) cost of implementation. The six projects were presented as alternatives named x1, x2, x3, x4, x5, and x6. Pairwise comparison matrices were created to evaluate the relative importance of the criteria and to show how each project can be addressed based on the previous criteria. The eigenvector was normalized for the entire criteria. The authors found that the combination of the two methods, AHP and ZOGP, provided a systematic approach to improving the selection of IS projects as one of the basic management activities in information system planning.

Shang and Sueyoshi [69] used AHP to help managers in the design and planning phase to select the most flexible manufacturing system (FMS). The authors developed a structured framework containing three parts: the AHP model, a simulation module, and a financial procedure. The framework was unified under a method called Data Envelopment Analysis (DEA). Each part had a certain mission. The AHP method was used to analyze the affect of the qualitative benefits at a strategic level, and was structured into three levels. The top of the hierarchy was the best MFG system in the long term. The descending levels contained four main criteria with associated sub-criteria: (1) flexibility, including economy of scope, backup capability, and design change accommodation; (2) learning, including experience and expertise, competitive advantage,
and leader of new technology; (3) capacity increment, including additional tools, saving for future investments, new product introduction; and (4) exposure to labor unrest, including direct labor union pressure and indirect labor union pressure. Three alternatives were proposed: (1) keep existing equipment, (2) FMS, and (3) new equipment. Pairwise comparisons of all criteria and sub-criteria were conducted and are shown in detail in [69]. The authors found the AHP method to be an efficient tool for evaluating the strategic benefits of implementing various manufacturing technologies, and to see how much flexibility is worth in technology.

3.4.5 Evaluation

Liang and Ren-wang [70] analyzed the profits of open source software and presented a model for pricing evaluation. This model used the AHP approach and divided the structure into three levels: (1) target level, (2) criteria level, and (3) program level. The target level contained the software total score. The criteria level included several evaluation criteria, such as practical degree, degree of innovation, risk cost, appearance factor, and software efforts. The program level included the object software and reference software. The experts provided qualitative judgments and weighed the data assigned for the criteria in the AHP.

The comparison matrix, consistency and eigenvalue calculation were successfully obtained. The authors concluded their study by developing a perfect AHP model to help in assessing the value of an open source software, especially in the Chinese context.

Xiu-qing [71] believes that success in an agile enterprise is linked with quick responses to the market. Business partners help in this matter, especially when sharing
resources and elevating the competitive power to the top. Thus, the author constructed a partner valuation model based on the AHP hierarchy model. The main goal was to reach the optimum selection for an agile partner. In this case, AHP was broken down into:

1. Objective hierarchy, which is finding the best partner selection.
2. Criteria hierarchy, including delivery date, quality, location, agility, information resources, and enterprise culture.
3. Sub-criteria hierarchy, which was inherited by the enterprise culture. The information resources criteria contained the following: cultural fusion, values, desires to share, quantity of information, information availability, and IT infrastructure.
4. Scheme hierarchy, which had various partner options (1, 2, 3, etc.)

A revised AHP calculation was used, starting from the lower hierarchy symmetric structure to the upper level of the hierarchy. The author then weighted the steps for each scheme to find the optimal one.

In the context of integration of automotive electronics, AHP was used alongside another method named ATAM (Architecture Tradeoff Analysis Method) [72]. Here, AHP helped in evaluating the integration decision concerning the physical connection of a vehicle electronic system. It also helped in estimating the quality attributes and analyzing the impact of various choices when prioritizing the desired proprieties of the system. The AHP and ATAM structures enabled the authors to choose the best integration strategies. Several functions and attributes were analyzed, including: safety critical, real time, control of the engine, drive train, driver interface, suspension, comfort, climate control, audio/video system, body, chassis, and infotainment. Moreover, the electronic hardware
components were discussed in detail to see how they functioned to interface the system that includes sensors, Electronic Control Units (ECUs), and wiring.

Wallin et al. [72] prioritized 16 scenarios obtained by having ATAM act as input for AHP. These scenarios were categorized into four groups based on interviews from different stakeholders: (1) safety, (2) reliability, (3) modifiability, and (4) serviceability. Each scenario was given a weight against each of the four main groups. Through a combination of AHP and ATAM, the authors finalized a feasible and suitable method to increase the accuracy of decisions in the context of integration of automotive electronics.

There has been research on system engineers trying to reduce traffic accidents caused by lorries. Jingliang Lv et al. [73] studied the reasons for traffic accidents and analyzed them using AHP. They presented in their research five main factors which generally cause the accidents and evaluated them using the AHP method. These factors were: (1) driver, (2) vehicle, (3) road, (4) weather and (5) the vehicle owner. AHP was capable of converting the human judgments into numerical values that could be compared across several alternatives. Five solutions were offered to be measured with the factors: (1) improving education, (2) stepping up punishment, (3) decreasing the vehicle checking cycle, and (4) improving the road. All the factors were weighted and the relative importance was determined by the assessment panel. The consistency check moved from the top level of the AHP structure to the bottom level. From the results, the authors found that based on the AHP structure weights, to reduce lorry accidents, the highest ranked solutions in order are: (1) improving education, (2) decreasing the vehicle checking cycle, (3) improving the road, and (4) stepping up punishment.
Davidson et al. [74] investigated the possibility of using AHP to evaluate architecture candidates, rather than using the ad-hoc fashion method for implementing a new architecture for the Multi-Agent System (MSA). Their investigation was applied to a case study that discussed the problem of load balancing and overload control in the networks.

The authors found that most of the current comparisons had been carried out in an unstructured manner. Thus, AHP was used as a structured method to evaluate the architecture’s attributes and provide a realistic comparison among the given architecture styles. AHP analyzed four MAS styles for dynamic resource allocation: (1) centralized synchronous architectures, (2) centralized asynchronous architectures, (3) distributed synchronous architecture, and (4) distributed asynchronous architecture. The MAS applications are various and have different requirements, so there is no perfect architecture for all MAS applications. As a result, the architecture styles in this study were evaluated with respect to six quality attributes as follows: (1) reactivity, (2) load balancing, (3) fairness, (4) utilization of resources, (5) responsiveness, and (6) communication overhead. The top-level goal in the AHP structure was to select the most appropriate architecture style among the four styles mentioned previously. All the architecture candidates were compared using the pairwise comparison process. The obtained values were normalized along with the ratio calculations. The authors took the results further to a real experiment in the area of telecommunication networks to create and maintain new types of service. Three types of agents were the main focus: (1) quantifiers, (2) allocators, and (3) distributors. By using the AHP method, the authors were able to determine the most appropriate architecture style for the MAS; this turned
out to be centralized asynchronous architectures. They also gain the following benefits from the AHP:

The ability to:

- “Quantify the differences in goodness of the candidate architectures according to the desired balance between the quality attributes.

- Weigh the different scenarios continuously.

- Easily add new instrumentations to increase the granularity of the evaluation.

- Easily add new architectures to evaluate.

- Easily add new evaluation criteria” [74].

Morera [75] indicated that the Commercial Off-The-Shelf (COTS) system is a common commercial system component that can be integrated with a current system to avoid building a new system from scratch. He raised three questions inherent to this matter: Which component should be used? Which component will fit better in the system? Which component should be trusted the most? The answers to these questions are very important, as they will reduce the risks which may result from inappropriate integration. Morera used AHP combined with DESMET methodology to establish an evaluation model for COTS in order to maximize the benefits of any component integrated with the software system.

DESMET methodology is an evolution method that helps the developers plan and execute an evaluation program. It originally resulted from collaborative projects in the United Kingdom. On the other hand, AHP is seen as multi-criteria decision method that structures problems using both quantitative and qualitative aspects. The process of
evaluating COTS used both the DESMET and AHP methods. In identifying the selection criteria, four factors were presented and evaluated as below:

1. Business factors such as historical records and COTS vendor recognition.
2. Financial factors such as COTS cost and upgrade costs.
3. Technical factors such as performance quality and safety.
4. Legal factors such as contracts and licenses.

In order to arrive at realistic findings, two different groups of people were involved in the weighting and evaluation process: technical and business people. The combination of the two methods (AHP and DESMET) provided a well-structured method for the company to choose the best components for its system. This approach was suggested by the European Software Institute under MOOSE (software engineering methodologies for embedded system).

Al-Ahmeri [76] wanted to improve the performance of Saudi industries by investigating the best implementation of computer-integrated manufacturing CIM. The CIM in this study included computer-aided design, computer-aided manufacturing, and computer numerical control. The author believed that implementing a new technology would elevate productivity and open new job opportunities in different fields, such as management information systems, requirement planning, and artificial intelligence. For this purpose, the author used AHP to select the proper CIM technologies with respect to several quantitative and qualitative factors. An extensive literature review was done for investigating the methods and tools used for CIM applications and selections. The author found that most of the approaches presented by the other researchers were incapable of
analyzing intangible factors. As a result, the AHP model was proposed to evaluate the CIM technologies and their impact on Saudi industries [76].

The data was collected from users and experts in more than 36 companies by using comprehensive questionnaires. Hence, statistical software (SPSS) was used to analyze such a huge data. The questionnaires targeted the 17 CIM technologies served: (1) CAD/CAE, (2) CAD/CAM, (3) modeling and simulation, (4) GT and CM, (5) FMS, (6) CNC/DNC, (7) AIS, (8) AMHS, (9) AS/RS, (10) MRP II, (11) ERP, (12) CAPP, (13) CAQ, (14) LAN and WAN, (15) EDI, (16) CIM, (17) AL, KBS, and ES.

The eight objectives selected were: (1) design cost, (2) cycle time, (3) tolerance, (4) productivities, (5) production and manufacturing cost, (6) machine utilization, (7) standardize the product design, (8) availability of real time information. All pairwise comparisons and consistency calculations were automatically achieved using MS Excel with VBA. With the AHP model, the author developed a reliable decision support system called CIMAHP that supported the tangible and intangible factors of CIM technologies. The final results showed that the CAD/CAM technology ranked highest with a score of 0.132. The second highest ranked was the CIM technology with a score of 0.116.

Yan Ge et al. [77] developed an AHP web service to evaluate online consumption behavior and its factors. The AHP web service was published in the UDDI registry, which allowed the resources and information to be shared conveniently. The AHP web service was structured into three levels: (1) TreeNode class, (2) AHP TreeNode class, and (3) Web Service class. The application had six steps to take the client from setting the objectives all the way to prioritizing and weighting the criteria and sub-criteria before finishing up with the AHP evaluation. The main factors affecting the online consumption
behavior presented in this study were: (1) products, (2) web sites, and (3) consumers. Each factor included some sub-factors. In order to evaluate these factors using the AHP web service, the client invoked the application remotely and displayed the results on the screen for a few seconds. Through this study, the authors provided a web service model based on the AHP method that could be considered as an applicable and independence platform.

Liebowitz [78] aimed to evaluate expert systems for the purpose of meeting their requirements and goals. The author reviewed some remarkable evaluation approaches such as: decision analyses, theory of measurable multi-attribute value functions, multi-attribute utility, and analytical hierarchy process. The AHP model was the best option because of the quantitative and qualitative factors of the expert systems evaluation. The author chose AHP to quantify the participants’ judgments and to help in addressing the important elements in the expert systems. The proposed evaluation in this study was derived from two studies by Boehm and Gaschnig [79,80]. The following list of criteria were developed by Boehm: (1) portability, (2) reliability, (3) efficiency, (4) human-engineering, (5) testability, (6) understandability, and (7) modifiability.

Following this, Gaschnig [79] developed evaluation criteria as follows: (1) quality of the system’s decisions, (2) correctness, (3) quality of human-computer interaction, (4) efficiency, and (5) cost. These sets evaluation criteria obtained from the two works were combined to create the following criteria: (1) ability to update, (2) ease of use, (3) hardware, (4) cost-effectiveness, discourse (input and output content), (5) quality of decision, and (6) design time. These criteria were structured in the AHP hierarchy to be
weighted and evaluated. Expert Choice was used to facilitate the complexity of the calculation. It also helped in visualizing the AHP structure with all the attributes.

The author applied AHP to evaluate the use of an expert system named READ or using human experts alone. READ is a software that enables users to identify the functional requirements for command management activities of NSAS. The criterion “quality of the decision “had the highest priority among the other criteria (0.443). The READ system was ranked better than using the human expert alternative with the following breakdown: (user1: 0.517, user2: 0.644, user3: 0.866). The option of using human experts was ranked as follows: (user1: 0.483, user2: 0.356, user3: 0.134). The author concluded that AHP was a powerful tool to measure the effectiveness of the expert system and illustrate the decision problems in an organized pattern [79,80].

Davis and Williams [81] presented a case study for using the AHP model to choose the right simulation software in a UK medium-sized engineering company. The evaluation of the simulation software packages proposed in this study satisfied the decision making in three areas: (1) strategic planning, (2) management control, and (3) operational control. The authors surveyed 14 packages over more than 100 software simulators.

In order to evaluate the software, the authors used a set of criteria and sub-criteria drawn from other researchers [82, 83, 84, 85] as follows: (1) cost, (2) comprehensiveness of the system, (3) integration with other systems, (4) documentation, (5) training, (6) ease of use, (7) hardware and installation, and (8) confidence related issues. Three of the criteria had sub-criteria, which are: (1) comprehensiveness of the system included flexibility, statistical facilities, graphical capabilities; (2) documentation included
instruction manual and reference manual; and (3) ease of use included expertise required, new user, regular user, and end user. The authors did not specify the alternatives. However, the letters A, B, C, D, and E were assumed to represent the list of software packages. The hierarchy structure for the simulation system selection problem was built, the factors were weighted, and relative importance was successfully measured with more than 178 comparisons made.

Douligeris and Pereira [86] had two main purposes in their research. The first purpose was to use the AHP method to evaluate the quality of the telecommunications service and choose the company which provided the best quality. The second purpose was to show how the AHP approach could be used to compare two high-speed network architectures, the Fiber Distributed Data Interface (FDDI) and Distributed Queue Dual Bus (DQDB). The authors first applied the AHP methodology to evaluate the quality of the Public Switched Telephone Network (PSTN). The criteria and sub-criteria affecting the quality of PSTN were presented as: (1) dialing ease, which included abandon before dial, abandon while dial, abandon due to poor transmission; (2) transmission reliability, which included SNR ratio, echoes, singing margin, interfering, cross talk; (3) call completion assurance, which included nature call, long hold time, short hold time; (4) switching and network, which included packet switching, circuit switching, software reliability, AI/Expert system, speech recognition, speech synthesis; (5) network security, which included authority, DES standard, public key crypt; (6) service restoration capability, which included prompt and effective service, no recurrence of problems, complaint answers; and (7) cost [86].
The criteria and sub-criteria made up the second and third levels of the AHP structure. The fourth level had the assumption of three companies (A, B, C). At the top of AHP was the goal: which of these company best fulfilled the customer needs by providing the highest quality of service.

In the second goal of the study was to demonstrate AHP as a useful tool for selecting the most suitable network technology. DQDB and FDDI were two high-speed networks presented to apply the AHP approach. Similarly, the authors investigated the most important factors affecting the decision selection process. Five criteria were discussed: (1) product performance, (2) reliability, (3) cost, (4) sales/service, and (5) customer satisfaction. Each of the criteria had several sub-criteria to break down the problem in detail. Questionnaires were sent out to a professional in the network field to evaluate the factors and sub-factors of the alternatives. All pairwise comparisons and priority calculations were performed. Through these studies, the authors provided a framework which can be used to arrive at the right decision to select the best telecommunication service. The rank for the DQDB was 1.036, which was approximately 2.5 times higher than the rank of FDDI0.416.

Akarte et al. [87] developed a web-based environment using the AHP model for casting supplier evaluation. The process of selecting the best supplier among several competitors is very difficult task and time-consuming problem, especially since outsourcing activities have a direct impact on cost and time constraints [87]. In Akarte’s study, there were 18 criteria for selecting the best supplier. These criteria were grouped into four main groups as follows: (1) product development capability, which included maximizing cast size, maximizing section thickness, casting complexity, software aid; (2)
manufacturing capability, which included pattern making, sand preparation, molding, core making, melting and pouring; (3) quality capability, which included heat treatment, machining, dimensional tolerance, surface roughness, testing facilities; and (4) cost and time, which included quality certification, quality awards, total casting cost, and sample delivery time. The supplier’s alternatives were not identified. The weights were assigned first to a group of criteria, followed by an individual criterion. All the criteria and sub-criteria were evaluated, normalized, and implemented in a prototype web-based system called InterCAST. The authors successfully developed a quantitative evaluation of casting suppliers using the AHP approach as a web-based system. They evaluated suppliers, A, B and C, based on the above criteria. The buyers gave relative weights for the supplier criteria based on the Request For Quotation (RFQ). For the criteria, the buyer weighted the software aid as the highest with (13.9%), followed by casting complexity (12%), pattern making (10.6%), testing facilities (9.8%) and minimum section thickness (8.6%). At the end of the evaluation, supplier B has the highest performance score (38.1%), then supplier A (33.8%) and finally supplier C (28.8%).

Last in this category, Cagno et al. [88] applied the AHP tithe field of auctions in order to achieve an accurate cost estimation and quantitative evaluation of the probability of winning bids for uncertainty process. The probabilistic type AHP was presented to address the uncertainty of the contractor judgments using the Monte Carlo approach. As an example, the authors structured the AHP hierarchy for designing and constructing a process plant in a developing country.

The AHP structure had three levels. At the top of the hierarchy was the competitive value of the bid. The second and third levels contained the following criteria
and sub criteria: (1) service level (delivery time, technical assistance, technical transfer), (2) plant performance (process technology, dependability, safety), (3) financial conditions (price, terms of payment, financial package, utilization of local vendors), and (4) contractual conditions (contract co-op, conformity documents, liquidate damages clause).

The alternatives bids assumed bid A, bid B, and bid C. Competitor A was presented as the “best” bid with the highest score (69.5%) and awarded the contract. Competitor B came in second, but way behind (28.2%). The last competitor was C is even further with a score of 2.3%.

3.4.6 Other purposes of using AHP

Barker and Shepperd [89] proposed using AHP to help software project managers make accurate predictions for both time and effort. The AHP method was presented as a solution tool for system estimation to overcome one of the major challenges in software development. Two test cases investigated and showed how the AHP approach could model effort prediction. Since effort prediction relies on historical software development data, the authors applied AHP to two real datasets. The first data set was collected from very large software central to BT’s core business. The second dataset was obtained from a similar organization. These data sets were estimated for effort by experts and contained many projects. The test case placed the effort attribute at the top of the AHP structure. Estimators were asked to determine which of two projects took more efforts. Pairwise comparison calculations and weights were then performed based on effort. At the end of the test cases, the authors found that the AHP method was well suited to guide estimators to achieve effort prediction in the presence of up to 40% erroneous comparisons.
Jung and Choi [90] used AHP for optimization models for the COTS software product among alternatives to consider the budget. Kunene and Petkov [91] used the AHP approach to investigate task decomposition to improve the process of idea-generating groups. The experiment used two groups of computer science students at the University of Natal in South Africa. It was a part of the requirements of the Decision Support Systems course. All the participants were familiar with analysis and design issues and IS methodologies.

The experimental task was obtained from a Systems Analysis and Design case study. The participants were divided into two groups of six members each. They were asked to design a system to support business Retail Company. Fifteen minutes were given to read and understand the task and forty minutes for generating idea solutions using Team Expert Choice as the problem-structuring environment. The quality of the ideas brought about by the two teams was measured by three criteria: total quality, mean quality, and the number of good ideas. The AHP model helps the researchers to come up with a significant finding: that task decomposition generates 40% more ideas than no decomposition.

Chou and Cheng [92] developed a systemic method using AHP to manage the image content and evaluate multiple criteria. Image retrieval and managing the image content are very important aspects in the field of memory information system. The image represented as a multi-dimensional vector was evaluated by the AHP technique through matrix-weightings. The imaged was classified into three layers that included objects, category codes, division codes, sections codes, and path codes. AHP was precisely used in this paper to assess the weights of the path codes in a vector. The users of the system
were instructed to conduct pairwise comparisons and set the relative importance between the two semantic descriptions. All the numerical values were arranged in a matrix and the calculations were automatically performed. When the image was presented, it was classified, and the category code was chosen using AHP. After that, the relative importance and pairwise comparisons were performed.

Gerdsri and Kocaoglu [93] applied the AHP approach as a part of a framework for a new technology called Technology Development Envelope using (TDE). This technology aimed to inject the road-mapping approach with more dynamic and flexible features to improve the development of technology strategies. AHP was used to evaluate the impact of technologies on the main objectives of an organization. All the comparative judgments and essential data were obtained by a group of experts within an organization that had already implemented a new technology and integrated them into their products.

The AHP model was carried out in three steps: (1) defining the company objective for technology evaluation with all associated factors and criteria; (2) providing comparative judgments for each pair criteria by experts; and (3) evaluating each technology by measuring their technological metrics impacts on the company. The seven criteria proposed to measure the effectiveness of technologies were as follows: (1) performance, (2) geometric, (3) reliability, (4) economic, (5) environmental compatibility, (6) serviceability and maintenance, and (7) flexibility [93]. The TDE approach was applied to a real example to help one of the leading computer server developers to maximize the benefits of using electronic cooling technologies. The authors found that AHP enhanced the dynamic and flexibility features of the TDE framework used in road-mapping technology. It also provided the managers with a clear vision and
understanding regarding where technologies can fit into their organization.

Moreover, Saaty [94] applied AHP to projects involving risk and uncertainty in investment. The author defined two types of uncertainty faced by decision makers: “(a) uncertainty about the occurrence of events, and (b) uncertainty about the range of judgments used to express preferences” [94]. He focused on studying the second type. Saaty showed how the AHP model could be used for risk estimation and alternative prioritization. Selecting the best portfolio was an example of the model’s application as presented in this study. The portfolio hierarchy had four main criteria: (1) return, which included interest, dividends, and capital appreciation; (2) low risk, which included diversification and low-perceived risk, and volatility of return; (3) tax benefits; and (4) liquidity, which included small transaction costs, ease of withdrawal, and the ability to make small additions. Five alternatives in the third level of the hierarchy were as follows: (1) tax free bonds, (2) securities, (3) saving accounts, (4) speculative stocks, and (5) blue chip stocks and bonds. From the AHP calculation, it was shown that the securities and saving accounts were ranked the highest (0.27), followed by tax-free bonds (0.16), and speculative stocks and blue chip and stocks bonds (0.15). The author proved that AHP was a helpful tool for dealing with the risk involved in priorities and probabilities in terms of understanding the complexity of the attributes needing estimation.

Finally, Kim and Whang [95] used AHP for forecasting the technological capabilities in an industry in order to determine the impact of certain technology elements on a product, and also to obtain relevant time series data. This method was applied to the real case of the Korean civil aircraft industry. The authors first measured the past technology index using questionnaires. Two aspects were taken into consideration: (1)
the time series technological index and (2) the technology index anchors for technological development stages. For the AHP hierarchy structure, the main goal was to evaluate the aircraft industry technology receptively with three criteria. These criteria were: (1) assembly/fabrication, which included airframe and auxiliary, engine and auxiliary, avionics auxiliary, raw materials; (2) aircraft design, which included system integration, aerodynamic design, structure design, engine design, flight control system design, system and equipment design; and (3) testing/evaluation, which included wind tunnel testing/evaluation, aerodynamic/structure testing/evaluation, engine testing/evaluation, flight control system testing/evaluation, and environmental testing/evaluation. The experts classified the element technologies and assigned priority weights for the technologies with respect to the time series data. For example, they classified the constituent technologies of the aircraft industry into three sub-fields: fabrication, design, and testing and evaluation. After computing the weights using AHP, the results were as follows: fabrication - 14.2%, design - 42.9%, and testing and evaluation - 42.9%.
3.4.7 Summary and charts for AHP in software system applications

Tables 3.10 and 3.11 show the number of AHP articles classified by their purposes and integrated methods.

Table 3.10 AHP purposes in software system development

<table>
<thead>
<tr>
<th>AHP Purposes</th>
<th>Number of Articles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision</td>
<td>5</td>
</tr>
<tr>
<td>Measurement and Assessment</td>
<td>5</td>
</tr>
<tr>
<td>Prioritization</td>
<td>6</td>
</tr>
<tr>
<td>Selection</td>
<td>24</td>
</tr>
<tr>
<td>Evaluation</td>
<td>12</td>
</tr>
<tr>
<td>Others</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 3.11 AHP and integrated methods in software development

<table>
<thead>
<tr>
<th>Integrated Method</th>
<th>Number of Articles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bayesian</td>
<td>1</td>
</tr>
<tr>
<td>GP</td>
<td>1</td>
</tr>
<tr>
<td>ATAM</td>
<td>1</td>
</tr>
<tr>
<td>DEA</td>
<td>1</td>
</tr>
<tr>
<td>DESMET</td>
<td>1</td>
</tr>
<tr>
<td>Delphi</td>
<td>1</td>
</tr>
<tr>
<td>TDE</td>
<td>1</td>
</tr>
<tr>
<td>Goal Programming</td>
<td>1</td>
</tr>
<tr>
<td>Total</td>
<td>8</td>
</tr>
</tbody>
</table>
Figure 3.10 shows how the AHP methodologies were distributed based on their purposes.

Figure 3.11 shows the percentage of AHP integrated methods and stand-alone AHP methodologies.
3.5 Summary of Lessons Learned:

1. AHP applications have been utilized in many fields, including logistics, manufacturing, government, high education, business, environment, militarily, agriculture, health-care, marketing, industry, personal, social, manufacturing, political, engineering, and software systems. This various uses of AHP proved their usefulness.

2. AHP has been used for different purposes such as selection, evaluation, planning and development, priority, ranking, decision, allocation, and forecasting. AHP has also been used as the sole method or integrated with other methods such as linear programming, integer linear programming, mixed integer programming, goal programming, quality function deployment, genetic algorithms, and artificial neural networks. These characteristics demonstrate the flexibility of AHP.

3. AHP has been used to make critical decisions and evaluations by big companies like British Airways [41], which deals with 1.2 million cargo shipments every year, and Marble Machine factory [59], which deals with a turnover of $4billion USD annually. These examples demonstrate the reliability of AHP.

4. The decision with AHP may not be the “absolute best” decision, but only the best decision among a limited set of alternatives. That is, unless one of the alternatives is the optimal solution to a problem, AHP will only be able to come up with the suboptimal best that it has been given.

5. AHP uses a good concept to reduce complexity when making a decision: breaking down a problem into components and studying the relationships among these components. This concept is presented in a hierarchy structure that has different
levels, and each item of a decision is placed in one level of the structure. Moreover, the weights for these items can be given by the decision makers to see their influence on the main goal.

6. The difference in value between any two criteria or any two alternatives can be easily quantified by introducing pairwise comparisons and using the scale number provided by Saaty.

7. The process of evaluating the alternatives and assigning weights to the criteria is easier than the process used by most other multiple-criteria decision analyses (MCDA).

8. AHP aims to optimize decision making, especially when the decision makers encounter situations that have a mix of qualitative factors (level of collaboration among the development team, the relationship between the pair programmers, confidence in expressing opinion and customer satisfaction) and quantitative factors (time, experience and cost). The notable advantage here is that AHP can utilize a scale for measuring the qualitative factors, called “intangible factors” in our study.
CHAPTER 4

APPLICABILITY OF XP IN EDUCATIONAL ENVIRONMENTS

In order to validate the results of any XP study, the case study’s environment has to be in line with XP principles and values. XP practices must be applicable in the environment setting in order to accomplish the required findings and observations.

This chapter explores the applicability of using XP in educational environments, specifically in post secondary institutions. It shows how the other researchers conducted studies and developed software using the XP methodology by students. This chapter helps researchers find ways of mitigating the threats to construct and internal validity. Observations, feedback, interviews and surveys can establish a chain of evidence for the possibility of practicing XP in the educational field. After investigating 14 XP empirical studies that were done in different universities and institutions, we concluded that XP is applicable in the educational environment albeit with challenges. Moreover, some researchers found that the productivity of students was only about 15%-20% less than other developers in industry [104].

The observations that other researchers published were considered in this study. More effort was put into overcoming the challenges that were encountered. This effort enhanced the setting of the XP development environment in this research.

In the following sections, we deeply investigate the difficulties that students face when adopting XP in class. Also, we highlight the important factors that affect XP’s adoption in education.

Almost all the XP practices that worked well in education exclude the 40 hours of work per week (as inapplicable). Students also found difficulties in applying some of the
practices in the context of the curriculums. Lack of experience, conflict in schedules, availability of the customer and on-site coach have significant impacts on the success of XP in the educational fields. The difficulty of implementing some XP practices raises a question about whether students applying XP will be successful. Will the students truly follow the XP rules or not? How can we adjust the educational setting in order to help students adopting the XP principles and values? By reviewing many XP studies that involve students in an educational environment, the answer for these questions will be provided.

4.1 Purpose of Applying XP in Education

Using XP in the educational environment has different goals. Yet, what do we mean by “educational environment”? In the literature review, we targeted students of a high education level, both undergraduate and graduate students, studying computer science or other relevant majors that teach some of the programming languages.

After reviewing 14 conducted studies, we found the purpose of applying XP in the educational environment is varied. It can be one of the following purposes:

1. Extreme programming or other agile methods might be one of the courses that taught in undergraduate or graduate program and apart of the degree requirement [96,97,98].

2. To investigate the effects of a specific practice of XP on the developers’ performance and development process [99].

3. To fulfill project requirements assigned for the fourth undergraduate student or even just a project for a computer or engineering course [100,101,102].
4. To find out the core problems and difficulties of teaching agile methods in an educational environment [103].

5. To evaluate some of the XP practices [103].

6. To teach the student how to collaborate with others as one team, and to show the benefits of working collectively [104].

7. To examine some important factors of XP such as: personality type, learning style, and skill levels and presenting their impact on the compatibility of pair programming [105].

8. To observe the impact of XP on the educational environment and get some experiences [101].

9. To compare the XP approach with other agile methods such as plan-driven development [106].

10. To establish creative ways to increase the pair compatibility applied to freshmen, undergraduate students or graduate students [107].

4.2 Overview of XP Empirical Studies

At the University of Karlsruhe, Matthias and Walter [96] experimented with some of the XP practices on a graduated course to observe the evolution of this method in the educational environment. Their study was focused on pair programming, iteration planning, testing, refactoring, and scalability. During only one semester, the authors came up with the following observations: first, the students enjoyed the pair programming and 43% of the participants learned from their partner, yet some of the students believed pair programming to be of little or no use. Second, designing in small increments needs
trained developers. Third, due to the lack of communication between the team members, the information exchange was not properly obtained. Fourth, the students found it very difficult to write the test cases before coding. However, 87% of the students said that the test case strengthened their confidence and improved the quality of the code.

The pair programming experiment was done at the International Islamic University Malaysia [99] to seek the benefits of the technique through a Java programming course for under graduate students. The study assumed that all students had similar levels of programming experience. A majority of students (96%) enjoyed pairing and 82% of the participants enhanced their understanding of the coding project and assignments.

Schneider and Johnston evaluated some of the XP practices [103] via students in the software engineering field in a learning environment. The study focused on pair programming, metaphor, on-site customer, and development cycle. When investigating pair programming, the authors found that most of the educational systems rely on a scheme where high marks are rewarded. As a result, matching pairs based only on the student’s marks could create problems balancing the strength of the team members. Also, they found that personality types and genders have a significant impact on the pair performance. In the development cycle, there is a problem with the short time spent working on the assignments by the teams, which differs in the educational environment and the industry. In the industry, it is normal to get 40 hours per week working on the project and the iteration can be every two weeks. The maximum time we get in education is 10-15 hours per week including the regular class and lab sessions. Moreover, the authors highlighted some factors that really affect the XP practices in the academic
environment such as the workplace, schedules, customer availability, and the student’s level of experience.

Webster University designed a graduate course to teach Agile Software Development following Object-Oriented programming in the curriculum [104]. The students explored some of the extreme programming aspects such as planning game, unit testing, refactoring, and pair programming. Peter Maher, the instructor of the course, played the role of the customer and observed the students’ behavior while experimenting with the XP process. The students had issues estimating the workload and finishing the stories on time. Also, they found that coming with a small design is not easy task. However, at the end of the class, the instructor observed a significant improvement on the all students’ work. About 80-90% of the course attendees provided positive comments in design simplicity, pair programming, ordering requirements, and customer relationship. About 10-20% of the students were confused and had concerns about the project velocity and time spent on the design. Finally, about 5-10% gave negative feedback regarding the testing. One of the important practices aimed at in teaching this course is working as a team and sharing knowledge. The students grasped the benefits of collaboration and reflected positively on that in their project.

A study was conducted at North Carolina State University to examine the ability of students to program in pairs based on personality type, learning style, skill level, programming self-esteem, work ethic, or time management preference. The author found that more than 93% of the pairs are compatible if they are similar in these factors. More than 1,350 students participated in this exam from different levels of education and experience: freshmen, undergraduate students, and graduate students [105].
Shippensburg University offers courses in software engineering that introduced extreme programming as a core part of the curriculum in the first two semesters [108]. The instructors require students to complete one project using two different methodologies. In the first semester, the students are required to use a waterfall process for the first release and in the second semester, to use the XP method for the second release. The goal of this change is to show the students extremes of the project development process and compare it with traditional process. Exercising the two methods, students learned that there is no optimal process that fits all the situations. The culture of the organization plays a significant role in choosing the appropriate process. The students also believed that using XP doesn’t mean having less discipline than using the waterfall process. The researchers observed that the students’ performance in the course has improved. Later on, the instructors made an official change to the software engineering curriculum by substituting the waterfall process with plan-driven development. Similarly, the students had to complete a project using plan-driven and extreme programming in two semesters. The students were more excited to start XP than the plan-driven process. At the end of the semester, the instructors came up with three important notes:

1. Switching the partners improves the pairs’ performance and helps to spread the knowledge.

2. The students showed negative results in their performance when using automated testing built by one of the faculty members.

3. The students were not successful designing incrementally even though they were taking a class in design patterns and refactoring.
Extreme programming was applied in the 4th year design project at the University of Calgary [100]. The project was implemented by a team of four students: three of them had no prior Java experience and only one member had some experience. Indeed, the team members were new to the XP approach. They team could not cover all the XP practices. However, they were able to experiment with pair programming, refactoring, incremental deliverables, unit testing, and configuration management. The challenges appeared in the early stage, and after four months the team quit the practice of pair programming, feeling that it was just a waste of time. Others indicated conflict in the pair schedules. The team also had difficulty in the planning and estimating phase. But because the group was very customer-centric, they produced a functional piece of software that met the customer’s desire. The students found that the use cases are a good way to provide a clear understanding of the expected code and also to display the customer’s requirements. In the refactoring process, the students spent a very long time adjusting the existing code and found it difficult to integrate the system. Yet, the team experienced the unit testing and ensured its benefits when the risk of errors was decreased. The students could have a total of three deliverables with only three defects. These defects were results of incorrect requirements capture. They also used the Java Concurrent Versions System as configuration management to reduce the risk of overwriting the code. This tool brought a confidence to the programmers and led the team to release the software.

Shippensburg University offers a 4-credit course that includes all the 12 XP practices [101]. The students had the challenges in five practices: pair programming, test-driven, the planning game, the iteration planning game, and refactoring. The students are assigned to a project that must be delivered within 14 weeks. Thirty students participated
and were divided into two teams of 15 students each. Most of the students worked in pairs in previous courses, so they were aware of the partnership aspect and comfortable with their partners. Nonetheless, the students faced a real problem in arranging extra time to pair outside of the class. Also, they did not switch the pairs regularly, which had negatively affected their productivity in the earlier stage. Even though the both teams performed some refactoring, one team enjoyed it more than the other. In the third iteration, one team had reserved about one third of its project velocity; this is a positive impact. At the end of the first iteration both teams failed to deliver functions for the planned stories. The teams attributed the failure to the lack of the customer’s involvement. This problem was diagnosed and solved by increasing the customer availability, which finally resulted in bridging the gap of understanding between the customer and the developers. Average of 50 % of the personal deliverables were computed in most of the iteration.

The most noticeable weakness was in using test-driven development especially for the GUI-related class. Wellington admitted, “the test driven development was not natural for our students” [101]. Eventually, the team could develop a successful project with a reasonable code quality and average of 152-production LOC per students.

Another XP experience was performed in a software engineering course at the Lund Institute of Technology in Sweden [97]. More than 107 students enrolled to the class and most of them had no prior experience of agile methods. Their programming skills were limited as well. The students had to work on a project during a 6-week period. The course was focused in some XP aspects such as planning game, small release, and on-site customer. There were some additional aspects such as on-site coach, and team in one
room, spike time, reflection, and documentation. Since the students were new to XP, the
instructors used some of the PhD students to coach the teams. The coaches kept track of
the team members and guided the students to apply the XP practices correctly. The
“Team in one room” aspect brought several benefits to the team in term of increasing
communications and having a common vision. Also, it facilitated coaches’ job: to spot
any problem easily.

Corel Hedin and Bendix [97] promoted developers having a specific time for
working individually, or “spike time”. It helped the team to read and learn about the
various practical issues. In addition, the team was occasionally assigned a meeting for
reflection on the current work to avoid any repetitive problems. The authors noted that
only around 10% of the students are women, but most of them had a positive experience.
One of the new practices implemented in this course was that the students were required
to have some documentation other than the code. This included the user stories, the
architectural description, and the technical report. The instructors noted that only one
team used refactoring tools in the project.

Another XP study was conducted in spring 2004 at the Metropolitan State College
in Denver [98]. Eighteen students participated in the one-semester project. They were
divided into three teams of six members each. The students had limited knowledge about
XP. They were introduced to the XP process through lectures, textbook discussion, in-
class exercises and video sessions. The instructor played two essential roles, one as a
customer and the second as an instructor. He switched between different colored hats to
distinguish the two roles. The students experimented with several practices and
techniques including pair programming, coding standard, simple design, refactoring,
testing, configuration management, and documentation. The students welcomed being paired and gave very positive feedback about pair programming in terms of writing great quality of code. Their feedback about the pair programming (on a Likert scale of 1 = Agree to 6 = Disagree) is as follows: pair programming was both easy to learn and useful (Agree = 1.83); it works well to create the software needed (Agree = 2.17); it helps to improve the code quality (Agree = 2.22); students like pair programming more than solo programming (Agree = 2.28). This is a good result, since some students had negative comments before the pair programming experience. Also, for the simplicity of the design, students followed the XP, that is, design only for the current iteration’s user stories and refactored their code frequently (Agree = 2.11). However, some students believed that the refactoring did not add any value to the system. Another point was that all the team members have taken their commitments seriously, which resulted in great collaboration. Here, the instructor found that the small size team was a major factor in having informative communication. The teams had no problem frequently releasing a piece of the software in a small iteration. In the testing phase, the students were also successful with the Junit testing. The major source of the problems was in the “small design” practice, the students struggled to understand the concept behind the simple design. However, the teams generally did well in most of other XP practices. The students responded favorably to the statement “the XP process was a good learning process in understanding many software engineering principles” (Agree = 2.44) [98]. At the end of the study, the authors believed that the XP model was not suitable for the software engineering course for undergraduate students.

In 2001 and 2002, Mugridge et al. taught XP [102] to undergraduate students for
the duration of two semesters and ran three different projects. More than 55 students participated in each project and two professors were involved to play the roles of the customers and coaches. After completing the three projects, the authors reached the following conclusions:

1. The amount of information related to the concepts and practices needs enough time to be learned.

2. Keeping tracks of the students, who were involved in multiple teams, is a very difficult task.

3. Some of the XP practices require advanced programming knowledge or having some experience in programming in order to do them properly. For example, doing the simple design and the writing the test before coding.

4. Having a real customer is not an option in most of the educational environment. Yet, acting as a customer required a willingness to be always available, which is also difficult for whoever is acting as a customer.

5. It is a challenge to ensure that the students are focused on high quality and well-factored code.

6. Extreme programming cannot be taught in one semester.

Melnik and Maurer [109] conducted an XP study at Southern Alberta Institute of Technology. The experience involved 45 students enrolled in the three different academic programs (Diploma, Bachelor’s, and Master’s). The authors determined the following perceptions: (1) the students were generally very enthusiastic about the extreme programming. About 91% of the students believed that using XP improved the productivity of small teams and 87% of them think that XP improves the quality of code.
Also, about 82% of the students said they would recommend using XP to their companies. (2) The majority liked the planning game and noticed a great improvement in the accuracy of their estimation. (3) In pair programming, some pairs faced a real problem caused by their weak and unskilled partners, so a suggestion was brought up by students to match the two partners based on the qualifications and experiences. Another problem regarding the pairing was caused by the conflict between the project schedule and the outside commitments the students had. However, about 84% of the students enjoyed pair programming and agreed it could improve software quality.

The test-first design was very confusing for most of the students. They believed that creating the test first was like working backwards. Only 58% of the students acknowledged doing it. The most surprising result was when the authors stated at the end that “there were no significant differences in the students’ perceptions between educational programs and experience” [109].

A study on pair programming conducted by NSF at the University of California indicated that the sources of difficulties in pairing were disparity in experience, scheduling conflicts, lack of understanding, and imbalance of driving and reviewing time [107]. Less than 5% of the pairs in each class could overcome the problem of time conflicts. The authors also found misconduct: some students submitted assignments with both partners’ names even if one of the partners did nothing. Investigation showed that those who cheated were focused on getting good marks rather than focusing on how to do pair programming in reality. Also, they found that the time pressure forced students to solve the problems separately, ignoring benefits of the pair programming concept. However, they observed that less than 2% of the experienced students were unwilling to
spend some time explaining the code or the material to their inexperienced partners. After 10 weeks’ observation, the instructors highlighted skill, experience, and culture as the most important factors to be taken into consideration when pairing.

Another study was conducted at the same institution to evaluate main contributors to increasing the compatibility of student pair programming. This study involved more than 361 undergraduate and graduate software engineering students. After three academic semesters, the research showed that the pairs are more compatible if students with “different personality types are grouped together, similar actual skill level are grouped together, similar perceived skill level group together, similar self-esteem are group together, same gender group together, similar ethnicity are group together”[107].

Positive results were received from the students in the Dundalk Institute of Technology that exercised the XP over two months [110]. Most activities were focused on the core development practices: simple design, pair programming, testing, code standard, collective ownership. Students enjoyed the experience of pair programming. They showed 15% of the code defect was reduced. About 58% of the students said that they would like to be paired with a stronger programmer, while 35% indicated that they would prefer to have an equally skilled partner. Also, the results showed about 56% of the students completed their test cases while the failure of the first exercise ranged from 0%-60%; this indicates that the students were challenged in performing this practice. All the students responded that they had applied the code standard. Some good feedback about the study was that 65% of students felt that the code belonged to both members. Only 16% stated that the code belonged to them.
Fourth year students in university of Sheffield applied the XP concept to a real system for a real customer [111]. The students had very good experience in programming, but none of them had heard of XP before. Each student was required to work 10-12 hours per week for the project. About 80 students participated and 3 business clients from Genesys were involved. The study focused on communicating with the client and capturing the real requirements to deliver high quality of software. The students dealt with two projects: an existing one that needed some major testing and debugging, and a new project started from scratch. The authors found out the students could carry out legitimate empirical experiments if the relationship with customer was strongly built.

4.3 Adoption Level of XP Practices

Most of studies reviewed in this chapter attempted to adopt the whole XP practice, but not all of them succeed in achieving this goal. Educational conditions, such as the number of students, the project duration, and the level of student experience, marked the boundary of the possible practices which can be applied. These conditions will be discussed as follows:

1. Number of participating students (team size). Team size is a main factor in communication. A large team can negatively affect communication and create more discussion among the team members. On other hand, a small team will allow effective collaboration and quick feedback. Most XP proponents strongly believe that smaller teams are more functional and productive compared to larger teams. However, defining the optimal number is a challenge. Jeff Sutherland says “There is plenty of data to show that team sizes over 7 result in significantly lower
productivity” [112]. While Appelo suggested that five is the optimal team size [113]. Several researchers and experiences showed that five is the best number for a productive team, but there is no consensus. Other important factors that can play roles in the team are the number of tasks, the communication channels, and the expectation of the team’s performance [114].

2. Project duration. The availability of time is a significant constraint in the educational environment. It directly affects both the team’s relation and the whole project process. It is not easy to do several missions, such as teaching XP, training students to follow the process, and applying XP practices, in one academic semester. However, most of the XP studies presented in this paper showed satisfactory results during this short period.

3. Level of experience. The students are categorized in one of the three educational levels: undergraduate, diploma, or graduate. The programming knowledge and problem-solving skills are part of the experience, but real-work involvement is more valuable than anything else.

In Table 4.12, we compare the 14 studies reviewed in this paper from four perspectives: adopted practices, number of students who participated, education level, and project duration.
<table>
<thead>
<tr>
<th>U</th>
<th>Applied XP practices</th>
<th>Participants</th>
<th>Experiences Level</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(1) Pair Programming (2) Iteration planning (3) Refactoring (4) Testing (5) Scalability.</td>
<td>12 Students (2 teams)</td>
<td>CS graduate Students had experience with the teamwork.</td>
<td>8 weeks.</td>
</tr>
<tr>
<td>2</td>
<td>(1) Pair Programming.</td>
<td>34 Students</td>
<td>- All students have similar levels of the experience.</td>
<td>One Academic Semester.</td>
</tr>
<tr>
<td>3</td>
<td>(1) Pair Programming (2) Testing</td>
<td>Not Specified</td>
<td>- Undergraduate Students.</td>
<td>2 Academic Semesters.</td>
</tr>
<tr>
<td>4</td>
<td>(1) Pair Programming (2) Refactoring (3) Simple Design (4) Incremental Deliverable (5) Use Case Requirement (6) Unit Testing (7) Configuration Management.</td>
<td>4 Students</td>
<td>- Undergraduate Students. - No experiences in Pairing.</td>
<td>4 Months.</td>
</tr>
<tr>
<td>6</td>
<td>(1) Pair Programming (2) Whole Team-Testing (3) Code Standard (4) Simple Design.</td>
<td>18 Students (Three teams)</td>
<td>- Limited prior knowledge of XP.</td>
<td>One Semester</td>
</tr>
<tr>
<td>7</td>
<td>(1) Unit Testing (2) Refactoring (3) Pair Programming (4) Metaphor (5) One-site Customer.</td>
<td>Not specified</td>
<td>- Graduate Students.</td>
<td>One semester</td>
</tr>
<tr>
<td>8</td>
<td>(1) Pair Programming (2) Planning game (4) Test-First Design (5) Collective code ownership (6) Continuous integration.</td>
<td>45 Students</td>
<td>- Various level of the experience starting from second year up to graduate students.</td>
<td>12 weeks.</td>
</tr>
<tr>
<td>9</td>
<td>(1) Pair Programming.</td>
<td>Not Specified</td>
<td>- Undergraduate students (freshman with no experience)</td>
<td>2 semesters</td>
</tr>
<tr>
<td>10</td>
<td>(1) Pair Programming.</td>
<td>316 Students</td>
<td>- Undergraduate Students. - Graduate Students.</td>
<td>3 Academic semesters.</td>
</tr>
<tr>
<td>11</td>
<td>(1) Planning Game (2) Small Release (3) On-Site Customer Additional practices: (4) On-Site Coach (5) Team in One Room (6) Spike Time (7) Documentation.</td>
<td>107 Students</td>
<td>- Undergraduate Students. - Most of them had no experiences.</td>
<td>Seven weeks</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2: Specified.</td>
<td>2: 12 weeks.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3: 70 Students</td>
<td>3: 6 weeks.</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>(1) Planning Process (2) Small Release (3) Pair programming (4) Metaphor (5) Simple design (6) Coding Standard (7) Continues Integration (8) Collective code ownership (9) Refactoring (10) Testing (11) One-Site Customer.</td>
<td>80 Students (5-6 teams)</td>
<td>- Undergraduate Students.</td>
<td>12 Weeks</td>
</tr>
</tbody>
</table>
4.4 Observations on XP Practices Applied by Students

This section briefly discusses the 12 main practices of XP and how they benefit the software development process as a whole. The challenges faced by students will be presented as well. The XP practices are: the planning process, small release, metaphor, simple design, testing, refactoring, pair programming, collective worship, continuous integration, 40-hour work week, on-site customer, and coding standards [26].

4.4.1 Planning process

Planning a release is a shared effort between the customer and the programmer. At the beginning of the project, the team will hold a release planning session to identify the features of the system. The stories are the most important valuable feature in this session. The word “story” is replaced by “requirement” in the waterfall method. It can be seen as the unit of the functions in XP [115]. In the educational environment the customer is needed to participate in the release planning. It might be a problem to have a real customer from outside. In most cases, the instructor plays the role of the customer, but he or she still needs to act as instructor for educational purposes. If the action of switching roles between the customer and the instructor does not go well, it will mislead the student and create obstacles to applying the XP process. The role of the customer in the planning
process is defining the user stories, determining the business value for the desired feature, and prioritizing the stories that will be implemented for the release.

The programmers (students) estimate each story (cost and time) and present the potential technical risks to be avoided. Both customer and programmer must establish a schedule for the release and the duration for the iteration. They answer two questions: on what should we work first? What will we be working on later? The students need some help from the instructor to overcome the following difficulties [97,104,109]:

1. Breaking big stories into smaller stories.
2. Knowing how many stories they will be able to do per iteration.
3. Estimating how long each story will take.
4. Responding to the customer’s changes about the requirements or the priority of the stories.
5. Measuring the team progress “velocity”.

Since this practice acquires some experience, the instructor has to provide a roadmap for students to reach their final destination. The instructor also has to switch his role to that of the customer and participate in forming stories, planning game, iteration planning, and estimating.

4.4.2 Small release

The XP team should release the iterative version of the software after it is tested. This release ranges from a day to a month. The customer expects to see the features he agreed to in the iteration. “Releasing frequently doesn’t mean setting aggressive deadlines. In fact, aggressive deadlines extend schedules rather than reducing them”[9].
The idea is to minimize functions and maximize confidence and trust. In other words, it is better to release a small set of functions that work properly and bring tangible value to the customer than to have more features, with their associated problems, and lose the customer’s trust.

After the customer sees the release, the feedback and comments can be given to the XP team before going on to the next iteration. The customer can cancel the work or change the priority order for the stories. Also, he might see unimportant features that can be removed and ask for the features he finds most valuable. The most difficult part for students in this exercises how to make small release regularly. However, they found the real benefits of regular small releases can be summarized as follows [97,101]:

1. They increase the trust and confidence between the customer and the programmers.
2. More feedback and comments will help improve the software.
3. They reduce the risk and pain that a team might have with the lengthy code and testing.
4. They reduce pressure on the team when the work gets close to the deadline.
5. They improve the estimating skills of the team members and speed up the velocity.
6. Continually testing the software reduces the defect rate.
7. They make the simple design step easier.
4.4.3 Metaphor

Explaining the system to somebody can be a difficult and repetitive task. The solution provided by the XP method is “metaphor”. It is possible to meet a person who says “Explain it to me like I’m a 4-year-old”[116].

The metaphor will help create a common vision of the system for the customer and the developers/programmers. It is a very powerful technique and is needed to bridge different backgrounds and levels of understanding. Moreover, the developers themselves will better their understanding of the system by giving examples and relating it to other areas. Additionally, the customer will be comfortable discussing the system and providing beneficial input about the functions.

In the educational environment, the instructor who plays the role of the customer normally has more experience and understanding than students about the project and the needed steps. The students need the metaphor, not by the customer, which is unusual. When the instructor acts as a customer, he must not switch his role to become a developer or coach. Otherwise, the practice of “metaphor” cannot be applied. The metaphor mainly works as a translator between two languages: programmer language and customer language. So, when the customer (instructor) becomes a programmer, it means there is no need for the metaphor. In most cases, this is what happens when the researchers attempt to apply the metaphor practice in the educational environment.

4.4.4 Simple design

The idea in this practice is to “design for today’s code with no extra design for tomorrow’s work”[26]. The system in XP is designed gradually. There is no need to
support the next iteration when you are working on features for the current iteration. Design enough for something that can be tested and running. Simple design is used to avoid the consequences brought on by complex design. For example, when a project has a very detailed design, it will need more testing, more refactoring, more time to fix bugs and defects, more feedback, and conflict understanding. Perfection is achieved when there is nothing left to take away, not when there is nothing more to add [9]. When students try to apply the idea of a simple design, they find it a challenge to answer two questions:

1. What is the simplest design we can come up with that could work?
2. Is the design ready to be modified easily any time?

From studies and experiences, simple design should have the following characteristics [26]:

1. Run all the tests.
2. No duplicated code.
3. Express developer intent.
4. The fewest possible classes and methods.

If the instructor can’t understand the students’ design, it usually means they are far away from the concept of simple design. Simple design is easy to modify and maintain. Yet, there are some difficulties brought by students such as [96,98]:

1. Confusing simple design and simplistic design. Some students provide a poor design and claim it is a simple design.
2. Distinguishing between design for today and design for tomorrow.
3. Avoiding duplication in code.
4. Thinking ahead about possible change when designing now.

5. Simple design requires continuous refactoring.

Overall, some students find that design is not a very sophisticated task, and some others think of it as a complicated job.

4.4.5 Collective code ownership

Students liked the idea that nobody has authority more than anyone else regarding any component of the system. This means each member of the team shares the same responsibility. Anyone can work on any part of the system at anytime. There are no methods or classes owners. No permission is required to make some changes in the code. However, the students found that collective work can be a problem when changes are made by students with less understanding of the written code.

4.4.6 Refactoring

Refactoring is the process of changing the design of code without changing its behavior. The developer has to finish all unit and integration tests before refactoring. In addition, changing the existing code requires courage and confidence that can be obtained by complete testing.

The students found the idea of “code smells” very helpful to answer the common question: when do you need to refactor? Fowler explained in detail the concept of code smells [118]. It means that a programmer can start feeling that the code smells bad or the partner finds understanding the code difficult. It does not always mean the code is bad. However, it can be visually determined by the programmer himself. There are many
indications that the code smells bad. Here they are, summarized [119]. Students found the following results of the refactoring practice [100]:

1. Enhances the design of the system.
2. Cleans the code and increase the readability.
3. Minimizes the code debt.
4. Eases the process of adding a new function.
5. Finds the bugs easily.
6. Makes the comments more meaningful.

Müller and Tichy refer the lack of refactoring to two factors: small size of the project and doing complete design rather than simple design [96].

4.4.7 Forty-Hour Work Week

Generally, people need to work overtime because they have to deliver what they promised to be done on the due date. As result, the quality and productivity drop dramatically.

Asking for more time is an indication of a big problem like poor planning, wrong estimation, or overcommitted deliverables. XP encourages the team development to stick to 40 hours per week or less. People are happy to go home on time with enough progress for the day as estimated before in the planning phase. Using this strategy improves the team’s skills and has positive impacts on other XP practices.

This practice is impossible to apply in the educational environment. Usually, students have an XP course in addition to other courses during the academic semester. If we assume that the XP class takes four hours per a week for regular lectures and labs and
six extra hours, that means the maximum we can get is 10-15 hours per a week [103]. Keep in mind that students are not willing to spend too much time on one class and ignoring other classes. It might be acceptable if students committed 10 hours, including the meetings and lab work, to working on the XP project. As a result, this time constraint must be taken into the consideration when: assigning the project to the students, providing the communication channels and setting up the lab hours.

The project that can be implemented by expert people using 40 hours per week should be different from a project implemented by students having 10 hours per week. The instructor can request that the students to keep touch and work with the other team members from home using Skype, Wiki, or other tools.

4.4.8 On-site customer

One of the essential requirements in XP is to have the customer available at all times to sit with the team, answer questions, and resolve disputes. Customers make decisions that might affect the business goals. Face to face communication with the customer is highly recommended in all the XP phases. Moreover, the customer has to take the commitment of his presence seriously to avoid any conflict in the feature between the developers and the business representatives. The XP customer is a part of the team and is involved in most of the development process. He or she is not just a normal customer [97].

In the educational environment, it is difficult to bring in a real customer from the outside, although not impossible, so the instructor usually plays the role of the customer [98].
The availability of the customer (instructor) outside the regular class is an issue, but it can be resolved. If the instructor is not willing to meet with a team, he should assign alternatives from the beginning of the project. Otherwise, failure will be a normal result for this action.

The students will need someone to act as customer for the following reasons [98]:

1. To make the decisions based on business goals.
2. To discuss the stories that will be included in each iteration.
3. To agree about the time scheduled for the release.
4. To write and prioritize the stories with developer assistance.
5. To tell the developers how he wants the system to be functioning.

Also, in the programming phase the customer is needed for the following reasons:

1. To provide the programmer with detailed thoughts about the feature and the system behavior.
2. To write the acceptance test before releasing the system.
3. To give their feedback and comments after the system features have been tested.
4. To give the green light to the developer to continue producing after reviewing the system.

### 4.4.9 Pair programming

Students can achieve numerous benefits when using pair programming, such as: learning from each other, fewer defects, better design quality, problem solving, peer pressure ensures timely delivery, less distraction, higher productivity, satisfaction, team building and communication, strong code ownership, sharing knowledge amongst team members, supporting self discipline, and providing feedback more quickly than schedule
inspection. Also, many mistakes get caught, the end defect content is statically lower, designs are better and code length shorter, students learn more about the system, the project ends up with multiple people understanding each piece of the system, and people learn to work together and talk more often [96,99,100,104,105,107].

However, most of the studies showed that there are common mistakes pairs make, such as:(1) absence of one of the members during the work time; (2) only one person does the job; and (3) the pairs do not switch.

The first thoughts and responses by students towards the concept of the pair programming are often one of the following [96,105]:

1. Pair programming is a waste of time. They wonder how the team will benefit from the idea of having a person sitting silent beside somebody coding.

2. Pair programming is a waste of effort. They think one job (programming) can be done by one person not two people. The other person can do another task.

3. Pair programming is not comfortable. Some students have their own style of coding and their way of handling the bugs, so they are not open to collaborating with another person who has a different style and thought.

4. The majority of students have never paired. However, even after pairing there is a portion of students who still prefer to code on their own.

5. Some of the students feel they are forced to pair but they are not really convinced.

6. The seniors think the work will be done slowly due to the juniors.

Therefore, instructors have to prepare answers for the for the challenges below:

1. How to convince students who don’t know anything about the XP to be paired and to believe the efficiency of the method.
2. How to structure the pairs, how to choose which two students will be paired together, how to make sure that the two pairs are matched (primarily through grades).

3. How to eliminate the impact of the personal characteristics conflicts between the pair.

4. How to evaluate the work properly.

5. How to overcome the schedule conflicts. Do the students need to pair all the time? Can they pair remotely?

From other perspectives, it is very important to focus on two main issues that affect the success of the pair programming [107]. These two issues are (1) matching the pairs and (2) setting up the workplace for the practice.

Matching the pairs is an essential step that can be performed by the instructors or the students themselves. Whoever takes on the responsibility of matching the pairs has to be aware of the following factors: personality types, experience, programming skills, availability for pairing, communication channel, gender, culture, language, and appearance.

As for the environment setting, most of the labs in the education system are designed for students to work individually, not in pairs. So, it is a matter to set up the pairing station, which is plays a significant role in XP generally, and pair programming specifically. Most of the labs in the universities have a straight desk, which is not comfortable for doing pair programming. So, the most proper workplace for pair programming should have:
1. Computer desk. The two people should sit beside each other comfortably; this means the computer desk must be customized for this purpose. Both developers get a straight angle to the screen. The flat desk will be preferable. Also, it is advisable to avoid any desk curve toward a person, which will make the working condition for paring very difficult.

2. Screen, keyboard, and mouse. It is good idea to have two screens mirroring the display combined with two keyboards and mice. Thus, each person feels like they have control over their own place. However, some developers prefer to work on one of each (screen, keyboard, and mouse).

4.4.10 Test driven development

Testing is about finding problems and presenting all the stories that were not implemented. The programmer and customer share the responsibility to get this practice done together. Test-driven development “is a rapid cycle of testing, coding, and refactoring. When adding a feature, a pair may perform dozens of these cycles, implementing and refining the software in baby steps until there is nothing left to add and nothing left to take away” [9]. TDD is one of the important XP practices that helps the programmer make sure that their coding acts exactly as they think it should. It is about producing small increments in the code quality and avoiding the massive list of outstanding bugs. Yet, some studies showed that students spend 50% of the project time on testing [115]. Other studies showed even more. At the Dundalk Institute of Technology, 95% rated time spent developing test cases as productive [110].
The testing normally follows the process of [26]: (1) Creating a test for the small feature. (2) Writing a simple code that makes the test pass. (3) Repeatedly, creating a test for another feature and write the very simple code. (4) Continuing until no story (problem) is left to be tested. Moreover, to accomplish great testing, the tester must avoid duplication, have a good design, choose correct names, and keep refactoring.

Different types of testing will be part of the XP testing phase as follows [100]:

1. Customer test (also called customer acceptance test)

   The customer writes the acceptance test story by story. It is created from the user story and each story can have more than one customer test. But don’t consider the user story as a full unit test. Therefore, the user story cannot be completed without an acceptance test. The customer will be in charge of reviewing and verifying the correctness of the test. The programmer makes sure the customer’s requirements are met and the system features are accepted. The benefits of this practice are to bring the customer’s ideas and thoughts to the team. In addition, he has to agree about the system behavior. An acceptance test also can be used as a measure for the development progress [108].

2. Integration test

   Any test that requires further setting for your current system, such as communicating with database or doing tasks across a network is an integration test. When students need to do more integration tests, it is a sign of having poor design.

3. Unit test

   This test is about testing the class and methods. It helps students to consider what needs to be done. The unit test must go 100% smoothly without a single error. If there is
an error, it must be fixed before adding another feature to be tested. Every single line of code must run correctly [120].

Here are the most notable observation found by applying this practice [96,97,98,100,101,108]:

1. Experienced testers are needed to work with the customer. A professional tester could make a great contribution to the team. In the educational environment, the customer (instructors) might have more experience than the programmers and testers (students).

2. The GUI test is very difficult and complex since most GUIs were not designed with testing in mind. One of the solutions, suggested by Wellington [101], is to make the GUI code as thin as possible and to provide an alternative input mechanism for testing the user interface.

3. The automating unit test can be very beneficial in some situations, but it can be more costly than the manual test. The problem with automated is the fact that the product changes, which tends to break the test. So, Marick investigated the answer for the question: “Is it really worthwhile to automate tests?” [121].

4. The team will have challenges if the customer is not be willing to be a core part of this task, which clearly indicates a broken relationship with the customer. However, if the customer cannot write the functional test because of his poor skills, further help should be available. The customer should give some examples in this matter, and then the programmers share the possible seniors and apply the customer’s comments. One of the common mistakes is that the programmers and
testers do all the work for the customer testing. To get good results in testing, the programmer should not test the code alone, because he knows how it works [9].

5. Students were surprised how small each increment can be. This technique is no indication of lack of experience. On the contrary, experienced programmers make smaller increments than freshmen.

6. Testing tasks, including the acceptance test, are a collective effort taken by the entire team.

7. Producing excellent tests can be a result of the combination between experienced testers and customer.

8. Good feedback will bring good testing.

9. The design will be influenced by the testing and improved until it meets the customer’s desire.

10. Unit test and integration test allow the developer the confidence to make changes anytime without fear.

11. Many students found the writing the test before the code is extremely difficult [122].

4.4.11 Coding standard

The only way to let the students understand each other’s code is to standardize it. Students in North Carolina State University were instructed to use the Sun Java coding standard. The feedback was very positive: 94% liked it and practiced it [120].
4.4.12 Continuous integration

In the traditional waterfall process, the integration is a hassling task that can consume a lot of time to reveal the design and code deficiencies. However, integration with XP is an ongoing task that keeps the software verified and tested. Students found this practice very helpful to catch the problems and fix them at an early stage. Students are encouraged to have an on-site coach to help them with integrating the system continually.

4.5 Influential Factors from the XP Studies

There are many influential factors that have a direct impact on when students at any post secondary school apply XP. In this section, we will discuss in more detail these factors as follows:

4.5.1 XP team

An extreme programming team in education will include the following people:

1. Students: The team in education is different from the team in industry. One of these differences is that in the education, we have no options of excluding any of these students registered in the XP class. All students will be part of the project. Students play the role of the developers (programmers and testers). The students must have a minimum level of programming skills in order to work as part of a productive team. Most of the XP studies are applied to computer science or software engineering students. Also, the XP exercise should not apply to students who are in the beginning of their study (first and second years). The picture of the
software development should be pre-taken in some courses before getting the students into an advanced topic in software process. The students’ schedules will make significant impact on the whole project since extra time is needed.

2. Instructor: Of the different responsibilities carried by the instructor, teaching is the highest priority. The XP teaching process will take various forms: lectures, labs, books and papers, etc. Above that, the teacher will supervise the students’ work for grading and educational purposes. Motivation and encouragement given by the instructor will deliver important energy to the students. The instructor must provide the students with a quick mode of communication for any questions. Another essential task related to the instructor’s mission is scaling the project, and watching information exchange among the students.

3. Customer: Who is the customer in the educational environment? Sometimes the instructor takes the role of the customer. That means a heavy load will be carried by the instructor, as a teacher and as a customer. Switching between the two missions can mislead the students. Having a real customer is the best choice. Any customer who is part of the team has to be fully involved in the whole development process. Also, the relation and trust with the other team members should be strong enough to give appropriate feedback and comments.

4. Coach: Students who have no experience need to be trained how to apply the XP practices and to follow its process properly. An on-site coach will benefit the students in this matter, at least in the earlier stages of the project.
4.5.2 Education objective

An instructor in an educational environment has the task of teaching students a subject in the curriculum and then grading the students based on their performance in the class. When conducting XP studies with students, the instructor will be challenged to balance the education goals and his involvement with work among the team members. The instructor has to present the XP concept, observe students, answer the questions, and work with the team, so switching between these roles might derail the educational aims. Moreover, XP encourages students by its practices (e.g. pair programming, code standard, and code ownership) to work as one team where everyone is responsible for the success of the project. Evaluating the students’ performance on a project done collectively is a very hard task [102,111].

On the other hand, the students want to learn, but getting high marks will also affect the truth of following the XP process. Studies showed that, in pair programming for example, some students let their partners do their jobs and submitted the assignment as if they had done it together, which is not true. Students might focus in passing the class rather than following the XP. Therefore, the results observed in the end of XP study might not be accurate.

4.5.3 XP practice issues

Investigating the XP practice and its cycle by students with educational constraints brought up three important issues:
1. Missing Practice: 40 hours per week is the only practice impossible to do in the educational environment. However, using other communication tools can help add time to the project.

2. Difficult Practice: The research shows that students had the most difficulty with simple design, pair programming, refactoring, and testing. More effort needs to be expended by all the team members, including the instructor and the coach, in order to help the students overcome the obstacles involved in these practices.

3. Development Cycle: XP is an iterative production. The team has to release a piece of working software in each iteration, which is quite different from the traditional process that students are aware of. The team will possibly fail in the initial iterations, but things will slowly progress. If the students cannot overcome the estimation and planning issues, they may keep failing to deliver any piece of the software on time. The XP principles need to be understood by students to motivate the application of the XP practices.

4.5.4 Assigned project

Any project assigned to the students must take into consideration:

1. Complexity: Students will be introduced to a new method (XP), so there is no need to make the project very complicated. The main aim is to follow the rules of XP, not test their skills in solving the problems.

2. Project time: The students have fixed time during the academic semester. It is impossible to request them to work on a class after it’s over.
3. Scope: The project definition, requirements, and constraints must clearly be explained to the students.

4.5.5 Communication and collaboration

Effective communication is a core value in agile development. It is the only way to keep the team informed and productive. In the educational environment, communication is one of the major problems due to the conflict of the students’ schedules. If students rely only on their attendance during regular class time, this short time will not be enough to build relationships among the team members. Students have to spend extra time working on the project outside of regular class time. Students’ attendance in the lab and other team meetings has to be taken seriously by team members. As mentioned previously, students will not work for one class 40 hours per week, so using other communication tools such as Skype, Wiki, email, and phone is recommended.

4.5.6 Environment setting

XP in the industry environment has more advantages due to the workplace setting, rather than the labs in the educational environment. Students will have to adjust the labs and some equipment’s location to facilitate face-to-face communication. A convenient workstation is the main factor encouraging students to work together and exchange information quickly. Also, social activities are needed to build relationships among the team members. The instructor should provide colorful cards for stories and watching the board with card movements.
Figure 4.12 illustrates the important elements when applying extreme programming in educational environment.

Figure 4.12 Essential elements in the XP empirical study in education
XP values and principles should be considered when conducting a study in the educational environment. The duration of the project is a very important constraint that determines which XP practices will be applied and which tasks will be delivered. Performing XP activities such as daily stand up meeting, prioritizing the requirements, and assigning tasks to the team members with roles are main the factors of success. The heart of the above diagram is the XP team which includes students, instructor, customer, and coach. The involvement and communication among the XP team members must be effective in order to produce high quality and satisfy the customer.

4.6 Considerable Finding for XP Studies by Students

We found good applicability of XP in the educational fields especially when having students as developers. Some of the XP practices were successfully adopted, while others were not. The students’ programming skills are a fundamental requirement. Additionally, the students’ experience will have a positive impact on the whole XP study. Pair programming was a new technique to many students but they found it enjoyable and useful. The researcher needs to keep the following in mind before performing XP studies in the educational environment:

1. An on-site customer will help the students overcome challenges with some of the new techniques and principles in a proper way. The students have to be trained at the beginning of the project; other XP materials (papers and books) should be provided for reading.

2. The instructor has to devote his time to working with students and providing detailed explanations for their questions. In case of taking the role of the
customer, the instructor has to be available all the time and fully involved in the project in each step. The skill involved in playing two roles (teaching and on-site customer) and scaling multiple teams will be essential.

3. Simple design, testing, and refactoring require collective work by all the team members, including the coach and instructor.

4. Communication and relation among the students must be strongly built to increase the productivity. Extra time for social activates will be beneficial in this matter.

5. Workplace and scheduling problems will remain core problems and difficult to adjust. However, any effort to accommodate the XP project in the educational setting will result in elevating the performance and quality.

6. XP study requires an additional person to observe the teams’ work and watch all the processes to write the notes and comments.

I have considered all these notes when I conducted my studies in the educational environment and I obtained positive results, which will be shown in the next chapters.
CHAPTER 5

CASE STUDIES SETUP

This chapter presents all the educational and industrial studies used in this thesis. First, we explain in detail the methodology and propositions for each XP practice which used AHP. Then, the two case studies that were conducted in the educational environment at University of Regina are introduced. In addition, this chapter contains other important information such as the students’ backgrounds, the duration of their studies, the development and communication tools, and the customer’s roles. Moreover, the information about the project that was assigned for the development is presented with the user stories written by the customers and students. At the end of this chapter, we show some information about the industrial case studies along with various projects details.

5.1 Methodology

As mentioned in chapter 1, the primary objective of this study is to investigate how AHP can be used with XP practices and where. How can AHP benefit these practices? In this work, several areas of XP are proposed to apply AHP. In each XP practice, AHP has a special goal and uses. To clarify, each practice can have its own research questions. The methodology used in this study is the case study methodology, which has been applied to two case studies in an academic environment and three case studies from the industry. The academic studies have applied AHP in 10 XP areas; two with the planning game, two with the simple design, two with the pair programming, and two with the refactoring, two with the testing, while the industrial studies have applied AHP in six XP areas: two each in the pair programming, refactoring and testing.
The five important components of case study design explained in [123,124,125] will be defined in chapter 5 and 6. These components are:

1. Research questions.
2. Research propositions.
3. Units of analysis.
4. The logic linking of the data to the propositions.
5. Criteria for interpreting the findings.

The research questions, propositions, units of analysis and criteria for interpreting the findings are presented in chapter 5. The logic linking of the data to the propositions is shown in chapter 6.

5.2 Designing the Case Studies

According to [126] and [127], formulating the research questions and propositions is very important step for designing a case study. In this research, they will be written in two formats. First, the research questions and propositions are written in a general format for “how” and “why” it is good to use AHP in XP. Second, due to the fact that AHP has various purposes and different uses in XP, the research questions and propositions are written specifically for each XP practice individually.

5.2.1 Research questions and propositions in a general format

a) Research questions:

There are three main questions in this research:

1. How does AHP help the XP team to achieve their goals in the adopted practice?
The goal of using AHP in XP is different from one practice to another, as will be explained in detail in the next sections.

2. What are the benefits that AHP brings to the XP practice itself?

3. How does AHP affect the team’s relationship and performance?

b) Study propositions:

The propositions in this study are based on the researcher’s knowledge of AHP and XP. According to [126] a researcher can use his/her beliefs or personal knowledge to define propositions. The propositions are similar to the hypothesis or theories that a researcher writes before conducting a study or experiment. However, the term “hypotheses” is mostly used with testable experiments and quantitative research [128, 129,130]. After conducting the study and finalizing the results, these propositions may be confirmed. The main propositions in the research are outlined below:

1. AHP captures important criteria and alternatives that need to be considered in each adopted XP practice. Also, AHP’s results show the order of importance for each alternative.

2. AHP facilitates the process of selection, ranking, prioritization and decision-making when it is needed in the XP practice.

3. AHP involves an informative discussion and improves communication among developers.

4. AHP resolves conflicting opinions among the XP team when applying a specific practice.
Specific research questions and propositions for each XP practice

AHP is introduced to planning game, pair programming, simple design, refactoring and testing for different purposes. As a result, the questions and propositions will be specified for each practice individually in the following sections.

5.2.2.1 Planning game

The primary objective of using AHP in the planning game practice is to investigate how AHP can be used to prioritize the user stories collectively? In a way that satisfies both consumers and developers. Another use of AHP is to rank the user story prioritization techniques and select the most suitable one. The following research questions provide a focus for our case study investigation:

1. How can AHP help the XP team prioritize the both customers’ and the developers’ user stories?
2. How can AHP help to select one of the prioritization techniques for the user stories among many tools and techniques?
3. How does AHP affect the developers’ relationship and performance?

The study propositions are outlined below:

1. AHP captures important criteria and alternatives that need to be considered when prioritizing the user stories and selecting the prioritization techniques.
2. AHP facilitates the process of prioritization in the planning game.
3. AHP involves an informative discussion and improves communication among developers.
4. AHP resolves conflicting opinions among the XP team when prioritizing the user stories and selecting one of the prioritization techniques.

5.2.2.2 Pair programming

The primary objective in the pair programming practice is to investigate how AHP can be used to form the best pairs in pair programming. Another use of AHP is to decide the matching rules when pairing. The following research questions provided a focus for our case study investigation:

1. How can AHP help the XP team to decide the best pairs based on specific criteria?
2. How can AHP help to decide the rule of matching that involves specific characteristics?
3. How does AHP affect the developers’ relationship and performance?

The study propositions are outlined below:

1. AHP captures important criteria and alternatives that need to be considered when deciding to pair two programmers and deciding the rule of the characteristics matching.
2. AHP facilitates the process of making decisions in pair programming.
3. AHP involves an informative discussion and improves the communication among the developers.
4. AHP resolves conflicting opinions among the developers when pairing.
5.2.2.3 Simple design

The primary objective of simple design practice is to investigate how AHP can be used when deciding the simple design tool for the XP team. Another possible use for AHP will be prioritizing the CRC cards. The following research questions provided a focus for our case study investigation:

1. How can AHP help the XP team to select the simple design tools based on specific criteria?
2. How can AHP help in ranking the CRC cards with specific criteria?
3. How does AHP affect the communication among the developers?

The study propositions are outlined below:

1. AHP captures important criteria and alternatives that need to be considered when selecting the simple design tool and ranking the CRC cards.
2. AHP facilitates the process of ranking and selection in simple design.
3. AHP involves an informative discussion and improves the communication among the developers.
4. AHP resolves conflicting opinions among the developers when practicing simple design and when defining the most important CRC cards.

5.2.2.4 Refactoring

The primary objective in the refactoring practice is to investigate how AHP can be used in ranking the internal and external code quality attributes. The following research questions provided a focus for our case study investigation:

1. How important is it to practice the refactoring using AHP?
2. How can AHP rank the refactoring methods based on specific criteria?
3. How can AHP affect the communication among the developers?
4. How can AHP help in saving the developers’ time while refactoring?

The methodology used in this study is four case studies: two case studies in an academic environment and two case studies in industrial environments with embedded units of analysis. The study propositions are outlined below:

1. AHP captures important criteria and alternatives that need to be considered when refactoring.
2. AHP facilitates the process of ranking and selection in refactoring.
3. AHP involves an informative discussion and improves the communication among the developers.
4. AHP provides a map to focus on the most important refactoring methods that increase the code quality.
5. AHP resolves conflicting opinions among the developers when practicing the refactoring and trying to find the smell code.

5.2.2.5 Testing

The primary objective in the testing practice is to investigate how AHP can be used in the automation decision. Another use of AHP is to rank the most important release indicators.

The following research questions provided a focus for our case study investigation:

1. How can AHP help the XP team decide if they should go with completely automated, semi-automated or manually operated testing.
2. How can AHP help to rank the release indicators?

3. How does AHP affect the developers’ relationship and performance?

The study propositions are outlined below:

1. AHP captures important criteria and alternatives that need to be considered when deciding the testing technique.

2. AHP facilitates the process of ranking and decision in testing.

3. AHP involves an informative discussion and improves the communication among the developers when performing testing.

5.3 Criteria for Interpretation

After identifying the propositions, the criteria for interpretation for the final results should be identified as well [126]. So, once the results are found and analyzed, we will need to compare them to the initial propositions in order to decide if there is match or not. So, the interpretation criteria as follow:

P1:

- Documents show that AHP presents the criteria and alternative for each adopted XP practice and shows the level of relations between the criteria and alternatives.
- The AHP’s results are shown clearly and in order for each alternative.

P2:

- Evidence indicates that the process of using AHP in each XP practice is easily understood.
P3:

- Evidence presents that AHP creates an environment of discussion among the XP team that helps to transfer knowledge and experience.

P4:

- Evidence shows that AHP carries the voice and opinion for everyone in the XP team and resolves the conflict in opinions in each use of AHP.

5.4 Unit of Analysis

According to [126] the unit of the analysis should be derived from the main research questions of the study. So, the main focus of this study is based on each XP practice used. For the planning game, the focus is to prioritize the user stories and select the best prioritization techniques. For the simple design, the focus is to decide which simple design tool should be selected and how can we rank the CRC cards using the AHP methodology. For the pair programming, the focus is to decide what are the best pairs and the matching rules of developers’ characteristics. For the refactoring, the goal is ranking the refactoring pattern based on internal and external qualities attributes. For the testing, the main aim is to decide the automation level and to rank the release indictors. So, the purpose of using AHP (decision, selection, prioritization and ranking) and the process of evaluation are units of analysis for this study. Also, the participants’ view of how AHP benefits each XP practice is another unit analysis. As result, this work is designed as multiple cases (embedded) with multiple units of analysis.
5.5 Data Collection and Sources

In the beginning of the each AHP-XP practice, we propose the criteria and the XP areas that we want to investigate in order examine the AHP tool’s ability and benefits. This data was collected from literature review and previous studies. To increase the validity of this study, data triangulation was obtained. The data sources in this study were:

1. Archival records, such as a study plan from the graduate students.
2. Questionnaires given to the participants when developing the XP project.
3. Questionnaires given to experts from industry.
4. Open-ended interviews with the participants.
5. Feedback from the customer.

However, the most important data source of this study was an XP project conducted at the University of Regina in a software design class in fall 2012. In addition, three companies initially participated in evaluating some of the XP practices and based on proposed criteria that affect the practice. Later on, two of the three companies involved in evaluating the AHP-XP results.

5.6 Questionnaires

After applying AHP to a specific XP practice, a questionnaire was employed in order to obtain the participants’ perceptions and experiences with AHP. The questionnaires consisted of two main parts. The first part included questions about the AHP as a decision and ranking tool used in the XP. The second part included questions about their perception on the direct benefit to the XP team and their satisfaction. Data
collected from the questionnaires is kept in the Microsoft Word files. The participants answered the questions and handed them back to the researcher.

5.7 Semi-Structured Interviews

After obtaining the AHP evaluation results for all the XP practices, we conducted semi-structured interviews with 12 of the students. We discussed themes and issues perceived as important in our observations and field notes. We divided the interviews in three parts. First, we asked open questions to obtain students’ general opinions about AHP. Second, we focused on AHP in each XP practice, finding out what the students liked what they disliked. Finally, we asked them where the best experience for AHP was among the all XP practices. During the interviews, data was collected in the form of handwritten notes. These notes were organized and categorized in a folder to ease the process of analyzing and to draw conclusions. This folder also contained separate sections for the researcher’s observations and perceptions during the research.

5.8 Field Notes

We also made extensive use of field notes that had been taken by the researcher during the implementation of the project, from the discussions among the students in the Wiki space used for communication, during the lectures, and from the customer feedback. These notes helped to draw the final results about using AHP for the XP development.
5.9 Academic Case Studies

To apply AHP to the proposed XP practices, we needed more than a semester to cover all the XP areas. All the case studies were run within the context of the extreme programming process. Moreover, they were conducted within software process and software design courses given by the Department of Software Engineering, University of Regina. These studies are from the courses taught in winter 2011 and fall 2012. The students were not new to the concept of XP. However, materials including books, white papers and class presentations are provided to ensure the correct implementation. Using AHP as a decision tool was new; so more explanation is given to gain more understanding.

The first case study was very short and conducted in 2011. It was mainly focused in prioritizing the user stories in planning game. Twelve students acted as developers and two professors acted as customers presenting the University of Regina project. The rest of the XP practices were experienced in a fully implemented project in fall 2012. The project was developed by 12 Master’s and PhD students and a real client from the industry. The project was developed within the industrial perspective. Two teams were required to develop real software using the XP process and inject AHP as a decision and ranking tool in some of the XP practices (planning game, pair programming, simple design, refactoring, and testing). Thus, the total number of participants in the two studies was 24 graduate students, two professors, and one real industrial client from a local company.

Most of the results collected are from the fall 2012 study due to the fact that majority of the AHP-XP activities occurred in that semester. All the details in following
sections will cover this part. While the details of the prioritization of user stories will be discussed in the planning game chapter in the user stories prioritization section.

5.9.1 Students’ background

The case study was conducted on graduate class of twelve students enrolled in ENSE 870:Advanced Software Design, a course offered by the Faculty of Engineering and Applied Science in the software systems engineering program at the University of Regina. The students’ backgrounds related to the study include several programming languages such as Java, C, C#, and ASP.net. The students also have some experience in developing databases and using SQL. However, the students have implemented projects previously using waterfall methodology.

5.9.2 Duration of the case study

The study was conducted within the timeframe of a single semester, fall 2013. Within the 870 ENSE software engineering course, the students were required to implement one project using XP as a development process method. During the development of the software, AHP was used in the proposed practices. The students were divided into two teams, six for each team. The timeframe for the project was divided into five iterations. Typically, students would spend two weeks on each iteration while conducting the project (10 weeks). Due to the limited time in the educational environment, students had to spend extra time outside the class period. The first three weeks at the beginning of the course was spent in introducing the objective of the class and required knowledge to pass this course.
5.9.3 Team formation

The 12 students participating in the case study were formed into two teams. Each two students in both teams were paired to apply the practice of pair programming. Prior to the team formation, information about the students and programming experiences was collected to balance the two teams. Each two students were paired to serve the investigation of some criteria regarding the pair programming technique and AHP will be in details discussed later on.

5.9.4 Customer involvement

In carrying out this case study, a real customer from one of the local company in Regina volunteered to be fully involved and available for the teams most of the time. The customer has been a project manager for more than 25 years and is CEO of INToo Solutions. He has many experiences with XP projects as well. The customer attended all the classes and presented with the teams during the development. In addition to the face-to-face communication, the customer was available through several communication tools used in the case study such as: email, Wikispace, Skype, and cellphone. Normally, both teams received the customer’s feedback and opinions at the beginning and the end of each iteration.

5.9.5. Customer’s role

The customer has adopted the roles proposed by Martin et al. [131] which are: driving the project, providing project requirements (user stories) and participating in prioritizing them, and involving in the quality control (acceptance testing). The only
external person to the project was the professor, so the customer maintained the trust both
of the development team and the wider business.

5.9.6 Prior the study

During the first three weeks of the study, the following topics were presented and
explained in the course: (1) extreme programming, which includes comparing the XP
with the waterfall methodology, the 12 XP practices, and the lifecycle of XP
development; (2) the Analytical Hierarchy Process, how to use it in general and how
injected it in several practices in the XP project; (3) the Issue Management System as a
project that was intended to be developed for that semester using the XP methodology
while applying the AHP where appropriate. Besides that, students were encouraged to
read additional papers and references, which would then be discussed in class.

5.9.7 Development tools

Team 1 chose to use Visual Studio 2012 and the Silverlight application for the
development environment with SQL Server 2012. C# was chosen as the programming
language. Direct modification of HTML for formatting the pages has been also used.
Similarly, the Team 2 chose to use Visual Studio 2010 professional, C#, ASP.NET
AJAX Control Toolkit, SQL Server 2008 & SQL Server 2012. For the simple design,
both teams used the CRC cards, whiteboard, and UML diagrams.
5.9.8 Communication tools

To overcome the time constraints in the educational environment, we encouraged the students to meet regularly face-to-face out of the class. Also, we created a page in Wikispace for sharing knowledge, questions, and discussion among the students and between the students and the customer. Also, the lecture time, cell phone, email, and SMS messages were helpful for communication as well. The two teams were required to report the number of hours spent for face-to-face meeting. Table 5.13 shows how many hours were spent per iteration.

<table>
<thead>
<tr>
<th></th>
<th>Iteration1</th>
<th>Iteration2</th>
<th>Iteration3</th>
<th>Iteration4</th>
<th>Iteration5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Team 1</td>
<td>18</td>
<td>14</td>
<td>25</td>
<td>23.5</td>
<td>35</td>
</tr>
<tr>
<td>Team 2</td>
<td>20</td>
<td>19.5</td>
<td>26.15</td>
<td>14</td>
<td>18.5</td>
</tr>
</tbody>
</table>

5.9.9 Project assigned to the participants

The students were required to develop an issue tracking system that would allow the user to keep track and record trouble reporting by the customer to improve the quality of customer services. The system is secure and only authorized users, based on five different roles for the owner company and its clients, can access the system. There should be just one owner company with a super administrator, managers and customer service employees. The client company may have two types of users: client company administrator and client company user.

The idea of this system was given by the customer who involved with us in this work. He is the CEO of INToo Solutions and owned a similar system. The customer
provided a list of the requirement, discussed with the students, and participated in converting these requirements into user stories. The following software requirements were given to the participants in the iteration:

1. The system should be a web software multi-tier application with SQL database as data repository.
2. System users are:
   a) Partner (clients, vendors) companies’ users who can create issue and confirm the close of an issue.
   b) Our company users with different levels of rights (create, edit, delete, read/view only, assign, etc.).
   c) Users’ access rights (authorization) should be defined in system configuration.
3. The system should handle user authentication via Microsoft Active Directory (AD).
4. The system should be able to handle users’ authorization with user-application groups inside or outside the AD.
5. The system should allow authorized client users to view their own company issue list.
6. The issue list should be equipped with an issue attributes filter.
7. Issue categories should be defined in the system configuration.
8. Customizable issue/problem work flow, i.e. multi status issue/problem resolution path.
9. Issue status should be defined in system configuration.
10. The system should be able to track issue status change history.
11. Issue priorities should be defined in system configuration.
12. The list’s columns should include the main issue attributes.
13. Issue’ attributes include:
   a) Issue unique ID
   b) Description
   c) Issue category
   d) Issue priority
   e) Issue dates include: incident date, authoring/reporting/creation date, fix date, and closing date.
   f) Issue status: open, reopen, assigned, fixed, and closed.
   g) Issue actors: reported by, authored by, assignees, lead assignee, and closed by.
   h) Send email alter flag.
   i) Related issues.
   j) Notes.
14. Each issue should have a list of notes defined by: authoring date, text, authored by.
15. The lead assignee can create issue dependency list.
16. Web system’s zones include: Internet, intranet, and extranet.
17. The system should access URL and the system security should identify the user zone based on the access rights.
18. The system can assign multiple assignees; each assignee can execute several jobs assigned by a supervisor.

19. The assignment/action attributes include:
   a) Assignee ID
   b) Assignee name
   c) Assignment/action type
   d) Description
   e) Assignment date
   f) Planned completion date
   g) Actual completion date
   h) Completion level
   i) Contribution level
   j) Notes – each assignment has a list of notes
   k) Each assignment has many jobs

20. Assignment’s job should have the following attributes:
   a) Job ID
   b) Job type
   c) Description
   d) Planned start date and actual start date
   e) Planned finish date and actual end date
   f) Number of work hours
   g) Charge rate per hour
   h) Notes
21. Ticket dependency support.

22. The system should support both internal software and hardware issue tracking and for external product support.

23. Reply tickets and sending notifications through emails.

24. The system should have a reporting module which provides assignees’ utilization, cost, issue statistics from clients’ point of view (P.O.V.) and company P.V.O.

25. The system should have configuration/administration module which manages

   a) The clients’ and vendors’ profiles which include:
      1. Company name
      2. Description
      3. Relationship type (client, vendor, value added partner).
      4. Business category
      5. HQ address and main contact including email
      6. Billing address and billing contact including email
      7. Number of employees
      8. Issue-authoring user list.

   b) Our company name, description, user list with their access rights

   c) Managing issue attributes (as in 13)

   d) Managing assignment/action types

   e) Managing assignment’s job types

   f) Users’ security / login profile (username and password).
5.9.10 User stories

In the section, all the user stories written by Team 1 and Team 2 are presented. The format of writing user stories explained by Mike Cohn was as follows:

“As a <type of user>, I want <some goal> so that <some reason>” [132]. The clause “so that” is left to optional.

First: User stories for Team 1

Iteration 1:

- User Story 1: As a system admin I want to edit owner company’s users so that I can efficiently manage users.
- User Story 2: As a system admin I want to authorize clients/vendors so that only authorized users can access the application.
- User Story 3: As a system admin I want to edit client/vendor users so that I can efficiently manage users.
- User Story 4: As a system admin I want to authorize owner company’s users so that the application can be secured.

Iteration 2:

- User Story 1: As an issue, I want to be displayed as a list so that I can be viewed in an organized way.
- User Story 2: As an issue, I want to be created using an editor so that I can be edited and created in the system.
- User Story 3: As an issue, I want to be defined using attributes so that proper information is available to the assignee to resolve the problem.
- User Story 4: As an issue, I want be categorized by my attributes so that I can be
informed about the status of the issue.

- User Story 5: As an issue, I want to be assigned so that I can be resolved.
- User Story 6: As an assignment, I want to be owned by an assignee so that I can be attended to/resolved.

**Iteration 3:**

- User Story 1: As a system admin, I want the client/vendor to authenticate so that only authorized users based on their role can access application.
- User Story 2: As a system admin, I want to limit access to the configuration panel so that only administrators can manage a company’s and it’s users’ profiles and protect data and application access.
- User Story 3: As an issue, I want to be visible to authenticated users so that information is protected and data is available to concerned stakeholders.
- User Story 4: As a system admin, I want to secure application navigation so that users only with required role can view the application.
- User Story 5: As a user, I want to create/update/delete an issue so my issue will be solved.
- User Story 6: As an admin user, I want to view issues so I can be informed about the status of my issue.
- User Story 7: As an admin user, I want to filter issues so I can manage my resources.
- User Story 8: As a system admin, I want to create a company user so that I can manage company profile.
- User Story 9: As a system admin, I should be able to create company’s users so
that I can efficiently manage users.

- User Story 10: As a system admin, I should be able to search Company’s users so that I can filter and see a specific company’s users.
- User Story 11: As a system admin, I should be able to edit Company’s users so that I can update a company’s user profile.
- User Story 12: As a system admin, I should be able create user names and passwords so that I can use authenticate users’ identities.
- User Story 13: As a system admin, I should be able edit user names and passwords so that I can manage user names and passwords for a company’s user.

*Iteration 4:*

- User Story 1: As a system admin, I want to get access to a company’s panel so that I can manage and reorganize all of my company’s detail.
- User Story 2: As a system admin, I want to add/modify the company’s information so that I can control the company the way I need to.
- User Story 3: As a system admin, I want to get access to a company user’s configuration panel so that I can efficiently manage that company user’s information.
- User Story 4: As a system admin, I want to filter company’s user by company’s name, first name and user’s role so that I can look for the specific user.
- User Story 5: As a system admin, I want each client/vendor ID to map to a single login ID so that so that all users login with their assigned login ID and the system can trace their steps in case of any issue.
- User Story 6: As a system admin, I want role-based access to the application so
that users can access authorized content and it can secure the application and 
information.

- User Story 7: As a user/manager, I want to generate an issue report so that I can 
  check the status of the current work.

- User Story 8: As a software developer, I want to write a well-documented code so 
  that it can be easily maintained in the future.

- User Story 9: As an admin/manager user of owner company, I want to 
  create/delete/update assignments so I can manage my resources.

- User Story 10: As an authorized user, I want to view assignments so I can be 
  informed.

- User Story 11: As an admin/manager of owner company, I want to 
  create/delete/update assignee(s) so I can manage my resources.

- User Story 12: As an authorized user, I want to view assignees so I can be 
  informed.

Second: User stories for Team 2

Iteration 1:

- User Story 1: As an owner company, I should be able to edit owner company 
  users.

- User Story 2: As an owner company, I should be able to authorize owner 
  company users.

- User Story 3: As an owner company, I should be able to edit client company 
  users.

- User Story 4: As an owner company, I should be able to authorize client company
users.

- User Story 5: As an owner company, I should be able to edit vendor company users.
- User Story 6: As an owner company, I should be able to authorize vendor company users.
- User Story 7: As an owner company, I should be able to edit client companies.
- User Story 8: As an owner company, I should be able to edit vendor companies.

**Iteration 2:**

- User Story 1: As a company, I should be able to author my issues.
- User Story 2: As an issue, I should have a list.
- User Story 3: As an issue, I should be able to be edited.
- User Story 4: As an issue, I should have multiple assignments.
- User Story 5: As an issue, I should be able to customize my attributes.
- User Story 6: As an assignment, I should be able to assign multiple jobs.

**Iteration 3:**

- User Story 1: As a user, I should be able to login.
- User Story 2: As a user, I should have authorization.
- User Story 3: As a company, I should be able to add users.

**Iteration 4:**

- User Story 1: As a user, I should be able to create a report which contains all the issues that meet the current issue filter criteria, so that I can see issues’ cost details.
• User Story 2: As a user, I should be able to create a report that contains all the
details of an issue, so that I can see assignments, jobs, and cost of that issue.
• User Story 3: As a user, I should be able to do company management, so that I
can add, edit, and delete companies.
• User Story 4: As a company, I should be able to customize attributes, so that I can
change attributes according to my company standards.

5.9.11 Summary for the user stories

The user stories and related tasks per iteration for both teams are summarized in
Tables 5.114, 5.15, 5.16 and 5.17.

Iteration 1

<table>
<thead>
<tr>
<th></th>
<th>User stories</th>
<th>Tasks defined</th>
<th>Tasks done</th>
<th>Tasks postponed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Team 1</td>
<td>4</td>
<td>12</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>Team 2</td>
<td>8</td>
<td>40</td>
<td>30</td>
<td>10</td>
</tr>
</tbody>
</table>

Iteration 2

<table>
<thead>
<tr>
<th></th>
<th>User stories</th>
<th>Tasks defined</th>
<th>Tasks done</th>
<th>Tasks postponed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Team 1</td>
<td>6</td>
<td>14</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>Team 2</td>
<td>6</td>
<td>30</td>
<td>28</td>
<td>2</td>
</tr>
</tbody>
</table>
Iteration 3

Table 5.16 User stories and tasks for iteration 3

<table>
<thead>
<tr>
<th></th>
<th>User stories</th>
<th>Tasks defined</th>
<th>Tasks done</th>
<th>Tasks postponed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Team 1</td>
<td>13</td>
<td>40</td>
<td>All</td>
<td>0</td>
</tr>
<tr>
<td>Team 2</td>
<td>3</td>
<td>13</td>
<td>All</td>
<td>0</td>
</tr>
</tbody>
</table>

Iteration 4

Table 5.17 User stories and tasks for iteration 4

<table>
<thead>
<tr>
<th></th>
<th>User stories</th>
<th>Tasks defined</th>
<th>Tasks done</th>
<th>Tasks postponed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Team 1</td>
<td>4</td>
<td>7</td>
<td>All</td>
<td>0</td>
</tr>
<tr>
<td>Team 2</td>
<td>4</td>
<td>19</td>
<td>All</td>
<td>0</td>
</tr>
</tbody>
</table>

5.9.12 Summary of the two educational projects:

In this section, some of the technical information per iteration for both teams in the educational environment is presented in tables 5.18 and 5.19.

Technical information for Team 1:

Table 5.18 Technical information per iteration for Team 1

<table>
<thead>
<tr>
<th></th>
<th>Iteration 0</th>
<th>Iteration 1</th>
<th>Iteration 2</th>
<th>Iteration 3</th>
<th>Iteration 4</th>
</tr>
</thead>
<tbody>
<tr>
<td># Class in packages</td>
<td>0</td>
<td>5</td>
<td>21</td>
<td>40</td>
<td>50</td>
</tr>
<tr>
<td># Methods in class</td>
<td>0</td>
<td>18</td>
<td>29</td>
<td>363</td>
<td>438</td>
</tr>
<tr>
<td>Line of Code in class (non-blank and non-comment)</td>
<td>0</td>
<td>144</td>
<td>2882</td>
<td>4499</td>
<td>5628</td>
</tr>
<tr>
<td># of attributes for a class in the model</td>
<td>0</td>
<td>12</td>
<td>50</td>
<td>517</td>
<td>711</td>
</tr>
<tr>
<td>DIT (Depth inheritance Tree)</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
Technical information for Team 2:

<table>
<thead>
<tr>
<th></th>
<th>Iteration 0</th>
<th>Iteration 1</th>
<th>Iteration 2</th>
<th>Iteration 3</th>
<th>Iteration 4</th>
</tr>
</thead>
<tbody>
<tr>
<td># Class in packages</td>
<td>0</td>
<td>22</td>
<td>40</td>
<td>40</td>
<td>44</td>
</tr>
<tr>
<td># Methods in class</td>
<td>0</td>
<td>45</td>
<td>116</td>
<td>116</td>
<td>129</td>
</tr>
<tr>
<td>Line of Code in class (non-blank and non-comment)</td>
<td>0</td>
<td>2000</td>
<td>5000</td>
<td>5020</td>
<td>5500</td>
</tr>
<tr>
<td># of attributes for a class in the model</td>
<td>0</td>
<td>150</td>
<td>639</td>
<td>700</td>
<td>706</td>
</tr>
<tr>
<td>DIT (Depth inheritance Tree)</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

5.10 Industrial Case Studies

Three companies participated in this study by using the AHP to evaluate six areas of the extreme programming practices. These areas are: (1) selecting the best pairs, (2) rules of matching two persons in pair programming, (3) ranking the refactoring techniques based on the internal quality attributes, (4) ranking the refactoring techniques based on the external quality attributes, (5) automation testing decision, and (6) ranking the release indicators. Two companies are located in Regina and one is located in Calgary. The total participants in AHP evaluation were 18 experts. Six people from each company have written their evaluation according to the current project they are working on. So, to preserve their anonymitys, A, B, and C replace the companies’ real names. Table 5.20 provides the following information for each company participated in the AHP evaluation:

- Average experience for the developers
- Information about the project that was used for the study
- Programming language.
- Iteration duration
- Development process.

<table>
<thead>
<tr>
<th></th>
<th>Company</th>
<th>Company B</th>
<th>Company C</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Average Experience</strong></td>
<td>14 years</td>
<td>10 years</td>
<td>12 years</td>
</tr>
<tr>
<td><strong>Application</strong></td>
<td>a Trouble Ticket System designed to manage all Telecommunication networks incidents, including Telephone, Data and Wireless networks.</td>
<td>Order Management System. The system captures and manages Customer’s Orders.</td>
<td>Gift Card System for HR system that allows employees to submit a recognition request for other employees and the request is send for approval to the employee supervisor. After that the request is sent to HR admin to review the request and process it.</td>
</tr>
<tr>
<td><strong>Programming Language</strong></td>
<td>JSF2(PrimeFaces, PrettyFaces, WebServices and Messaging) for front-end, EJB3 for Business tier and Hibernate/JPA/Oracle for back-end</td>
<td>Java6.0</td>
<td>C#</td>
</tr>
<tr>
<td><strong>Iteration duration</strong></td>
<td>Two months</td>
<td>2 weeks</td>
<td>A month</td>
</tr>
<tr>
<td><strong>Development Process</strong></td>
<td>Agile/ Scrum</td>
<td>Partial XP / Scrum</td>
<td>Partial XP</td>
</tr>
</tbody>
</table>
5.10.1 Summary for the three industrial projects

Similarly to the educational environment, tables 5.21, 5.22 and 5.23 show the technical information per iteration for the three companies.

**Technical information for company A:**

<table>
<thead>
<tr>
<th></th>
<th>Iteration 1</th>
<th>Iteration 2</th>
<th>Iteration 3</th>
<th>Iteration 4</th>
<th>Iteration 5</th>
</tr>
</thead>
<tbody>
<tr>
<td># Class in packages</td>
<td>308</td>
<td>421</td>
<td>479</td>
<td>577</td>
<td>381</td>
</tr>
<tr>
<td># Methods in class</td>
<td>1293</td>
<td>1818</td>
<td>1437</td>
<td>2308</td>
<td>1686</td>
</tr>
<tr>
<td>Line of Code in class</td>
<td>12628</td>
<td>22734</td>
<td>21555</td>
<td>32312</td>
<td>19352</td>
</tr>
<tr>
<td># of attributes for a class in the model</td>
<td>1817</td>
<td>2123</td>
<td>2868</td>
<td>3411</td>
<td>1978</td>
</tr>
<tr>
<td>DIT (Depth inheritance Tree)</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

**Technical information for company B:**

<table>
<thead>
<tr>
<th></th>
<th>Iteration 1</th>
<th>Iteration 2</th>
<th>Iteration 3</th>
<th>Iteration 4</th>
<th>Iteration 5</th>
</tr>
</thead>
<tbody>
<tr>
<td># Class in packages</td>
<td>15</td>
<td>12</td>
<td>8</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td># Methods in class</td>
<td>27</td>
<td>23</td>
<td>21</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>Line of Code in class</td>
<td>445</td>
<td>420</td>
<td>370</td>
<td>350</td>
<td>350</td>
</tr>
<tr>
<td># of attributes for a class in the model</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>DIT (Depth inheritance Tree)</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

**Technical information for company C:**

<table>
<thead>
<tr>
<th></th>
<th>Iteration 1</th>
<th>Iteration 2</th>
<th>Iteration 3</th>
<th>Iteration 4</th>
<th>Iteration 5</th>
</tr>
</thead>
<tbody>
<tr>
<td># Class in packages</td>
<td>90</td>
<td>85</td>
<td>84</td>
<td>82</td>
<td>80</td>
</tr>
<tr>
<td># Methods in class</td>
<td>560</td>
<td>530</td>
<td>480</td>
<td>430</td>
<td>388</td>
</tr>
<tr>
<td>Line of Code in class</td>
<td>7600</td>
<td>7300</td>
<td>7150</td>
<td>6800</td>
<td>6381</td>
</tr>
<tr>
<td># of attributes for a class in the model</td>
<td>660</td>
<td>621</td>
<td>601</td>
<td>588</td>
<td>590</td>
</tr>
<tr>
<td>DIT (Depth inheritance Tree)</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
The researcher has visited these companies several times and met with the developers and team leaders to explain the purpose of the study, and to collect data and feedback from the real industries. The experts have used their knowledge and experience to evaluate the proposed areas of XP using AHP. All their results will be presented in the next chapters. After seeing AHP’s results, we had to go back to them again to validate the results. Two companies agreed to do the refactoring practice and the third company was unable to do so due to the shortage of time.
CHAPTER 6
APPLYING AHP WITH XP PRACTICES

This chapter is divided into five main sections. Each section presents a proposed XP practice that can benefit from using AHP. Planning game, simple design, pair programming, refactoring, and testing are the heart of XP development and open up the potential for injecting AHP in some specific areas with regards to its needs.

6.1 Planning Game

This section describes two proposed uses of AHP in XP planning practice, which is based on certain criteria. Activities central to the planning game such as user stories and prioritization techniques are introduced. This section shows a way of prioritizing XP user stories collectively using AHP. Also, it presents AHP-prioritization techniques that work to select the best one. Hierarchy structures for AHP that include specific criteria are drawn, and alternatives are suggested. The findings of the educational cases studies and their results are presented and discussed.

6.1.1 User stories overview

User stories are “short descriptions of functionality told from the perspective of a user that are valuable to either a user of the software or the customer of the software” [133].
• They are narrative texts that focus primarily on the value derived from using the systems, rather than focusing on their limitations. They provide detailed descriptions of these system’s behaviours to illustrate the advantages that the user can gain from using them.

• The user story is not a detailed specification of the software requirements. It is written in a collaborative environment facilitated by agile team. On the other hand, the user story promotes face-to-face communication between the business executives and the developers, which corresponds to one of the most important Agile Manifestos.

• The user stories are far more easily prioritized, and work to facilitate the estimations of factors such as time, cost, and risk during the planning stages.

The traditional requirement (TR) is written from the perspective of the system and focuses primarily on its operation while putting aside factors such as the user interface or the economics [134]. The main concern in TR is to understand the scope of what the system can accomplish by taking into consideration the constraints that it is subject to.

• It is difficult to write good traditional requirements due to the high expectations placed upon their completeness and formality. The International Institute of Business Analysts (IIBA) emphasizes certain criteria that can lead to good requirements. For instance, the requirements must be complete, testable, consistent with each other, and clearly written [135].

• TR is slow to respond to changes and expensive with regards to the contract. A simple change by the user results in a painful process for the developers.
• The level of detailed requirements with the “big requirements up front” leads to the creation of thousands of independent “shall statements”. The interdependencies between the requirements make prioritization a critical obstacle.

6.1.2 Factors affecting the requirements prioritization

Requirements can be prioritized according to several factors. There is no consensus about which of these factors is more important in determining the prioritization of the system requirements. However, almost all the factors taken into consideration aim to reach the most appropriate decision in order to maximize the value delivered to the customer.

Bakalova et al. [136] suggested that seven aspects must be considered when making decisions about the prioritization of the requirements: the context of the project, criteria regarding prioritization, the effort required to estimate measurements regarding size, associated dependencies, input from the developers, learning experiences, and the external changes. For the prioritization criteria, the authors focused on the business value estimated by the customer.

Hoff et al. [137] discussed the factors that affect the decision when determining the system requirement priorities, such as the cost-benefit to the organization, impact of maintenance, complexity and the performance effects. Also, the authors presented other factors obtained from a research was written by Wohlin [138] which need to be considered when deciding which requirements are more important and need to be implemented earlier. For example, in order to make a
decision, stakeholders need the delivery data, the requirements of the issuer and a
cost-benefit analysis of the development. Additionally, Hoff added four important
points which need to be taken into account when taking a decision on prioritization,
which were- the possible impact to the organization, probability of success, prior
ersors addressed, and testability. The authors completed a comprehensive survey to
determine which factors were the most important. They found that these were the
cost-dependencies, complexity, and delivery data/schedule have been marked as the
most important factors when prioritizing requirements for implementation

Somerville and Sawyer [139] prioritized requirements based on an approach
that represents information about the system’s requirements from different
perspectives. This viewpoint is often associated with the stakeholders, operating
environment, and system’s domain. A stakeholder can be either an individual, or an
organization that is interested in the system. Also, the future environment plays a
significant role in prioritization since a system always will be installed in a different
environment that introduces other data controls and components, which in turn
impose certain requirements. Additionally, the domain of the application tends to act
as constraint upon the system’s requirements.

Davis [140] used Triage as means of evaluating those requirements, which
possess a higher priority level with regards to factors such as time, resources
available, and the interdependency. Lutowski [141] prioritized the requirements
according to the importance or immediacy of need.

Bhoem [142] considered the cost of implementing a requirement as the most
important factor within prioritization. The cost encompasses a variety of elements
such as the technical environment, complexity, quality, timeframe, documentation, availability of reusable software, participant competencies, and stable requirements.

Favaro [143] used a cost-benefit analysis to look at those requirements which added value to the customer. He discussed three ways of realizing this impact: (1) creating reusable requirements; (2) using iterations with limited requirements which needed to be developed only for the current iteration; and (3) evaluating the financial impact of strategic decisions.

Berander and Andrews [144] analyzed the literature and found that there were common elements in strategies for prioritizing requirements such as penalty, cost, time, risk and volatility. Moreover, the author indicated that other aspects such as the financial benefits, competitors, release theme, strategic benefit, competence/resource, and lastly the ability to sell, should not be neglected.

Karlsson, and Ryan [145] used a cost-value approach for prioritizing the requirements considering the value and the cost as two primary factors in evaluating the system requirements.

Carlshamre et al. [146] found that only 20% of the requirements functioned independently, while more than 75% of the requirements were interdependent. He suggested considering the requirement interdependencies as a primary factor to be used when prioritizing, while others have suggested that the importance and volatility should also be considered [147].

In Wieger’s approach [148] the requirements were prioritized based on the system benefits, penalties, technical costs and risk associated with the implementation. When using agile methodology, Patel and Ramachandran [149]
prioritized story cards (agile requirement) according to business functionality, customer priority, core value, market values, implementation cost, and business risk. Wohlin and Aurum [150] introduced 13 different criteria arranged into three categories:

1. Business (External Customers/Market): This set includes criteria such as the competitors, requirement’s issuer, stakeholder’s priority of requirement and requirement volatility.

2. Management: This set includes the criteria of cost-benefits, recourses and time for delivery.

3. System (Development/Maintenance Personnel): This set includes the criteria of system impact, complexity, requirement dependences and maintenance.

Firesmith [151] looked at different dimensions for prioritizing requirements such as personal preferences, business value, harm avoidance, risk, cost, difficulty, time to market, requirements stability, dependencies, type of requirement, legal mandate, frequency of use, reusability.

Table 6.24 summarizes the factors affecting the software requirements, which were obtained from prior studies.
Table 6.24 Factors that affect the software requirements

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cost</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Risk</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Complexity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resources</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Domain</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dependences</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Market value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reusability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Urgency</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Possibility of success</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operating environment</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6.1.3 A Collective activity approach

Most XP supporters claim that the customers should always prioritize the user stories. We have conducted two short case studies to show the real need for developer participation during the prioritization as well as the need for customer inputs.

In the first study, we investigated the behavior of the developers and the customers when the prioritization is left to the customers. In the second one, we proposed a better way to prioritize the user stories through several insights from the developers and the customers’ representatives.

6.1.4 Uses of AHP in planning game

The AHP method can assist the XP development team in prioritizing user stories. Another use for the AHP in the planning game practice would be to select the most suitable prioritization techniques. Both AHP uses have been applied in the two test cases and results were collected.

6.1.5 Firstly: Prioritizing the user stories

The first use of AHP is to assemble the user stories in an accessible form to be used by customers and developers. The proposed user stories and the factors affecting prioritization will be discussed as follows.

6.1.5.1 Domain of study

This study is the only study that was performed in a different semester, as opposed to the rest of the studies in this thesis. It was conducted as a course activity
in the Software Process Class for graduate students taught in the winter of 2012, at the University of Regina. The participants were 12 masters students and two professors. It was carried out over three sessions of a day. The study was divided into three major sessions: an introduction to the user stories practice and the AHP method, the case studies, and a calculation and analysis of the results. The participants in this study had different levels of programming experience, and possessed an adequate familiarity with XP practices. The assignment was explained briefly as follow: The University of Regina wanted to develop an iPhone/iPad application for its website. This application could aid all potential users visiting the U of R page. As a primary stage development, the participants were required to write the user stories and prioritize them using the AHP method.

**First study**

The students and professors were introduced to their roles and divided into two teams of 6 members each. The students acted as developers and the two professors acted as customers. Each customer was assigned to one team. The customers and developers in each team sat down together and discussed the system requirements. The developers asked questions about the desired features and the customers answered. The communication was positive and the user stories became more detailed. Both teams were required to come up with the most important user stories for the proposed project. Five stories from each team were selected to be ranked by the AHP.

The following user stories were written:
1. As a student, I would like to be able to access and display events in different categories.

2. As a faculty member, I would like to be able to review the details of course requirements.

3. As a student, I would like to be able to access the university map that shows my current location in order to improve my mobility.

4. As a student, I would like to check my schedule of classes for the day to organize my events and to be able to study better.

5. As a member of the staff, I would like to post information about the opening and closing hours of the cafeterias, menu items, and the availability of food specials.

For this study, the customers were requested to prioritize the user stories based on five criteria obtained from table 6.24. These criteria were:

1. Value: the benefits that a customer could derive from the system.

2. Cost: implementation expenses including money and labor.

3. Risk: possible risks that should span scheduling risk, unforeseen expenses, and functionality risks.

4. Uncertainty: the unknown technical needs or implementation ability.

5. Complexity: the degree of the task complication.

The AHP structure for the user stories with the five criteria can be seen in Figure 6.13.
6.1.5.2 First findings and results

Each developer individually evaluated the user stories on the criteria mentioned previously. In doing so, they ignored their identity as developers. When they evaluated the user stories with regards to the value, they asked the customers which user story they found most valuable. However, it was noticed that when it came to any criteria other than value, the developers proved to be more knowledgeable. The summaries for the observations from the first study are as follows:

1. As mentioned previously, prioritizing the user stories is a duty left to the customer. However, in the real world, the customers can only evaluate the stories based on the value, while the other criteria such as cost, risk, uncertainty, and complexity remained beyond the customer’s capability and knowledge. A discussion between developers (students) and the customers (professors) led to a case study, which was performed by the developers. At this point, the customers gave up their roles and the developers switched
between acting as customers when evaluating the value and acting again as
developers when considering the other criteria.

2. The study did not stop when errors were made; we let the participants
independently realize the differences between the two roles, and what should
be prioritized when evaluating each criterion. As result, the developers took
over the role of the customer and applied AHP to rank the importance of the
user stories. They only consulted with the customers when they evaluated the
stories in the context of the value criteria.

3. The developers found it difficult to distinguish between the criteria of
uncertainty and risk. They attributed the same significance to both of them
during the process.

4. The urgency of the user story was considered as very important criteria for
the customers. We added these criteria in the second study.

5. The level of programming experience and the skills acquired by the
participants had a significant impact on the evaluation of the stories.

Second study

To resolve the confusion in the roles between the customers and the
developers, we requested that the developers and customers participate together in
ranking the user stories according to the new criteria assigned to each of them. The
students were divided into two groups: six students acted as developers and the other
six acted as customers.
With regards to the user stories, the customers were requested to write eight user stories and prioritize them. The user stories were as follow:

1. As a professor, I would like to be able to access URcourses to upload my assignments.
2. As a professor, I would like to be able to access the grading page to fill in the students’ grades.
3. As a student, I would like to be able to register for courses in the coming semester.
4. As a general user, I would like to be able to access the courses calendar to see any possible conflicts between courses.
5. As a student, I would like to be able to access my financial services account in order to review transactions.
6. As a student, I would like to be able to submit an assignment so that my professor can receive it on time.
7. As a professor, I would like to be able to access the library to help me find a new publication.
8. As a student, I would like to be able to view the merchandise at the bookstore to purchase new textbooks.

In second study, the goal was to prioritize according to two perspectives: (1) The developers’ perspective and (2) The customer’s perspective.

The developers considered three important factors: risk, cost of implementation, and complexity. On the other hand, the customers had to prioritize
the user stories according to the value and urgency. The process of prioritization followed the following steps:

1. Developers sat together to discuss the user stories, by asking each other three questions: which user story is associated with high risks? Which user story would be more costly to implement? And which user story is more complex in its scope? After that, they applied AHP to evaluate these criteria and prioritize the stories.

2. Similarly, the customers sat down together to discuss the user stories that they considered the most relevant and the most valuable.

3. When both the developers and customers finished the evaluation, and received the results of the AHP calculation, the customers reviewed both results and decided the direction of the final prioritization.

The communication was positive and the project requirements became more detailed. All the user stories were written on small cards. Many open discussions were conducted between the developers and the customers. The discussion focused on the importance of the derived user stories. The customers explained why they wanted to have them in the system. Some developers gave advice regarding the stories that they felt the customers liked but were unnecessary. In each iteration of XP software development, there were a number of stories that needed to be implemented. The question in this case was which of these user stories should be implemented first. The answer that came to mind was- the user stories most valuable to the customer. However, this answer did not take into account other factors that needed to be considered before prioritizing the user stories. These factors come from
the developer’s side. They were important for the customers to know, and could influence their decisions. This is because the customers could have chosen any user story to be implemented early in the process, but if they were informed about the risks and the costs associated with that story, they might choose to either postpone it to a later stage of the development, or cancel it. For this reason, mandating that the customer should always prioritize the user stories is not always the best option.
The idea in this instance is to let the developers participate in prioritizing the stories due to their prior knowledge. At the same time, customers should still be able to prioritize the stories according to their needs. Though the developers make sure to inform customers about the economic impact of each user story, they still leave the final decision to them. When both the developers and customers finish the evaluation...
and have the results of the AHP calculation at hand, the customers review both results and decide the final prioritization. The process for the case study 2 is shown in Figure 6.14.

In Figures 6.15 and 6.16, we structured two models of AHP for the user stories from both the developers and customer’s perspectives. The customers’ AHP structure takes into account two main criteria: value and urgency. On the other hand, the developers’ AHP structure takes into account the risk, cost and complexity. The customers compared the value and the relevancy using the Saaty scale. After that, they compared the user stories to each other with regards to their criteria. Similarly, the developers conferred and compared the risk, cost, and complexity. Finally, they analyzed the user stories according to their own criteria to inform the customers, before deciding which user stories should be implemented in the earliest iterations.

![Diagram](image.png)

*Figure 6.15 AHP structure for prioritizing user stories from the customer’s perspective*
6.1.5.3 Second finding and results

Each of the developers and customers individually evaluated the user stories based on the criteria mentioned earlier. The MakeitRational software [152] was used to calculate the aggregation results for the two teams collectively.

From the developers’ perspective, the priority of the implementation of a user story is accorded based on the factors of the risk, cost, and the complexity criteria. These were evaluated as follows: First: US3 (23.50), Second: US5 (19.01), Third: US7 (17.86), Forth: US2 (13.70), Fifth: US6 (10.38), Sixth: US4 (6.04), Seventh: US1 (6.69), Eighth: US8 (3.38). See table 6.25.
On the other hand, the order of user stories from the customer’s perspective was based on the factors of value and urgency. It was evaluated as follows: First: US3 (35.54), Second: US4 (15.83), Third: US6 (10.72), Forth: US5 (9.68), Fifth: US1 (10.39.598), Sixth: US2 (8.86), Seventh: US7 (6.72), Eighth: US8 (3.52). See table 6.26.

The methodology for the importance of each of the criteria relative to each other for the developers and customers is shown in Figures 6.17 and 6.18.
Figure 6.17 Importance of the criteria for the customers in the prioritization of user stories.

Figure 6.18 Importance of the criteria for the developers in the prioritization of user stories.
If we order the user stories by considering each criterion individually, the list will be different, as shown in tables 6.27.

Table 6.27 Ranking the user stories considering each criterion individually

<table>
<thead>
<tr>
<th>Stories</th>
<th>Risk</th>
<th>Stories</th>
<th>Cost</th>
<th>Stories</th>
<th>Complexity</th>
<th>Stories</th>
<th>Urgency</th>
<th>Stories</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>US4</td>
<td>3.41</td>
<td>US1</td>
<td>5.00</td>
<td>US8</td>
<td>4.68</td>
<td>US7</td>
<td>7.23</td>
<td>US7</td>
<td>5.55</td>
</tr>
<tr>
<td>US8</td>
<td>3.31</td>
<td>US8</td>
<td>2.87</td>
<td>US4</td>
<td>3.51</td>
<td>US8</td>
<td>4.34</td>
<td>US8</td>
<td>2.91</td>
</tr>
</tbody>
</table>

Note: even though AHP was a helpful tool in evaluating the prioritization of the user stories, the teams raised their concerns regarding the time spent on classifying the user stories according to the proposed criteria.
6.1.6 Secondly: Ranking the prioritization tools

This section introduces a ranking approach to help stakeholders select the best technique for prioritizing their user stories.

6.1.6.1 Introduction

To avoid conflicting opinions, stakeholders usually adopt a well-established prioritization technique used by XP teams such as: Ping Pong Ball [153], Pair-Wise analysis [154], Weighted criteria analysis [155,156], Dot Voting [157], Binary Search Tree [158,159], Ranking [148], Numeral Assignment Technique [160,161], Requirements Triage [162,163], Wieger’s matrix approach [164,165], Quality Function Deployment [166,167,168], Bucket technique, [169], Cumulative voting [170,171], Round the group prioritization [172] Theory-W [173], and Theme Scoring [174].

However, this adoption process is usually not formalized. In this section, AHP is used as a well-structured multi-criteria decision making tool to help XP software development teams rank six prioritization techniques: the 100-Dollar Test (Cumulative Voting), MoSCow, Top-Ten Requirements, Kano Model, Theme Screening, Relative Weighting.

6.1.6.2 Prioritization techniques

There are several methods for prioritizing the system requirements. The six most commonly used ones can be summarized as follows:
1. **The 100-Dollar Test (Cumulative Voting)**

This is a simple and straightforward technique described by Leffingwell and Widrig [175] used in prioritizing requirements, where each stakeholder gets 100 imaginary units (money, hours, etc.) to distribute as he sees fit among the given requirements. If the requirements are too big, it is recommended that stakeholders use the value units of 1,000, 10,000, 100,000, etc., to have more freedom in the process of prioritization [176]. After distributing the units on the requirements, stakeholders calculate the total for each requirement and rank them according to the highest totals, in order to indicate their relative importance.

2. **MoSCoW**

This is one of the easiest methods for prioritization originating from the Dynamic Software Development Method (DSDM) [177]. The requirements are classified into four groups depending on the importance of the functional requirements [178]:

- **M:** MUST have this. It is the highest priority, and without it the project is considered a failure.
- **S:** SHOULD have this if possible. It is expected to have it. The customer won’t be very satisfied if it is missing. However, a team cannot attribute a project’s failure to its absence.
- **C:** COULD have this as long as it doesn’t affect anything else. The priority is moderate.
- **W:** WON’T have it this time but WOULD like to in the future.
This technique helps developers to understand what the customer actually wants. The problem with this method is its failure to distinguish between the terms “must and should” since both terms indicate that the customers require the feature.

3. **Top-Ten Requirements**

In this approach, the stakeholders select their top ten requirements without assigning an internal order for specific requirements. This is done in order to avoid conflict between stakeholders, in the event that there is a difference in opinions regarding priorities. Any stakeholder can have more than ten primary requirements, but taking the average is a large challenge for this method due to the fact that it is possible to have some stakeholders who would not obtain any of their top priorities [179].

4. **Kano Model**

This method was established for product development by Noriako Kano in 1987 to order the requirements into five categories defined according to two questions asked about every requirement [180]: (1) functional question: “How do you feel if this feature is present?” (2) Dysfunctional question: “How do you feel if this feature is NOT present?”

The customer has to choose one of the five possible options for the answers [181]:

1. I like it.
2. I expect it.
3. I’m neutral.
4. I can tolerate it.
5. I dislike it.
5. Themes Screening

This is a very easy technique employed when stakeholders have many relevant user stories that need to be grouped together. While writing the stories, stakeholders eliminate similar or redundant stories. Then they follow the steps below [182]:

1. Identify 5-9 (approximately) selection criteria that are important in prioritizing the themes.
2. Identify a baseline that is approved and understood by all the team members.
3. Compare each theme to the baseline theme for each criterion. Use “+” for themes that rank “better than” the baseline theme, “-” for themes that rank “worse than” the baseline theme and “0” for themes that rank “equal” to the baseline theme.
4. Calculate the “Net Score” by summing up all the plusses and minuses. Rank as number one the theme that received the highest Net Score.

6. Relative Weighting

This technique evaluates each requirement based on the impact that its presence or its absence has on the project. A scale from 0 to 9 is identified for each requirement, 0 being a low effect and 9 being a high effect. The stakeholder will give every feature a value for its presence as well as a penalty for its absence and estimate the cost of its implementation. The priority is calculated by dividing the total value by the total cost to generate a prioritization indicator [182].
6.1.6.3 Proposed criteria for ranking the prioritization techniques

To rank each technique, it is necessary to determine the most important criteria that influence the participants while they are choosing a prioritization process. The resulting criteria will be compared to each other in order to reach the goal. Finally, the prioritization techniques will be compared to each of the criteria [31]. In this investigation, we propose four criteria that emerged during the course of the study, but the method described in this work can be applied to any set of criteria. The criteria shown below are simply illustrative of the prioritization method.

1. Simplicity: What is the simplest prioritization technique in terms of comprehension and implementation?

2. Time: Which one of these techniques is the most timely to explain and implement?

3. Accuracy: Which one of these techniques produces the most accurate results?

4. Collaboration: Which one of these techniques aids communication and collaboration among members of the team?

6.1.6.4 AHP- prioritization techniques structure

The first step in the analytic hierarchy process is to structure the problem as a hierarchy that has three levels. The primary level is the main objective: ranking the prioritization techniques; the secondary level is the criteria: simplicity, time, accuracy, and collaboration; the tertiary level is composed of the alternatives: 100-Dollar, Top-Ten, Kano Model, Them Screening, Relative weighting, MoSCow. Figure 6.19 illustrates the AHP structure for the problem.
6.1.6.5 Pairwise comparison process for the prioritization techniques

All the students had to use the prioritization techniques throughout the project in order to understand the advantages and disadvantages of each technique. Then they were required to evaluate these tools based on certain criteria. For this purpose, sheets of paper with appropriate AHP tables were handed to all students in order to facilitate the process of the comparison.

The students first compared the criteria among each other using the Saaty scale from 1-9. The participants asked the following questions:

- Which is more important: simplicity or time and by how much?
- Which is more important: simplicity or accuracy and by how much?
- Which is more important: simplicity or collaboration and by how much?
- Which is more important: time or accuracy and by how much?
- Which is more important: time or collaboration and by how much?
- Which is more important: accuracy or collaboration and by how much?
After finishing the criteria comparisons, the students had to evaluate all the prioritization techniques according to each criterion, every time they embarked on the process. These examples are as follows:

- In term of the simplicity, which is simplest 100-Dollar or Top-Ten and by how much is simple?

Similarly, all the following comparisons conducted based on each criterion:

(100-Dollar $\times$ Kano), (100-Dollar $\times$ Theme Screening), (100-Dollar $\times$ Relative Weighting), (100-Dollar $\times$ MoSCoW), (Top-Ten $\times$ Kano), (Top-Ten $\times$ Theme Screening). (Top-Ten $\times$ Relative Weighting), (Top-Ten $\times$ MoSCoW), (Kano $\times$ Theme Screening), (Kano $\times$ Relative Weighting), (Kano $\times$ MoSCoW), (Theme Screening $\times$ Relative Weighting), (Theme Screening $\times$ MoSCoW), (Relative Weighting, $\times$ MoSCoW).

The same questions and comparisons were repeated until all the prioritization techniques evaluated based on each criterion.

6.1.6.6 AHP evaluation results for the prioritization techniques

Each student individually evaluated the prioritization techniques based on the criteria mentioned earlier. The Expert Choice software [37] was used to calculate the aggregation results for the entire two teams collectively.

For Team 1, the ranking for the prioritization techniques based on all criteria, i.e. simplicity, time, accuracy and collaboration, was summarized as follows. First: relative weighting (24.39); Second: MoSCoW (20.38); Third: Theme Screening (17.70); Forth: Kano (15.81); Fifth: Top-Ten (12.75); Sixth: 100-Dollar. Table 6.28
summarizes the results. Figure 6.20 shows the importance of each criterion as follows: accuracy (48.04), simplicity (20.01), collaboration (18.00), and time (13.95).

<table>
<thead>
<tr>
<th>Technique</th>
<th>Scores</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relative Weighting</td>
<td>24.39 %</td>
</tr>
<tr>
<td>MoScoW</td>
<td>20.38 %</td>
</tr>
<tr>
<td>Them Screening</td>
<td>17.70 %</td>
</tr>
<tr>
<td>Kano</td>
<td>15.81 %</td>
</tr>
<tr>
<td>Top-Ten</td>
<td>12.75 %</td>
</tr>
<tr>
<td>100-Dollar</td>
<td>8.97 %</td>
</tr>
</tbody>
</table>

Table 6.28 Prioritization techniques ranking for Team 1

The ranking for the prioritization techniques by Team 2 is summarized as follows: First: Relative Weighting (32.67); Second: To-Ten (26.12); Third: MoScoW (15.44); Fourth: Them Screening (15.35); Fifth: 100-Dollar (7.15); Sixth: Kano (3.27). Table 6.29 summarizes the total results.

Figure 6.21 shows the importance of each criterion as follow: accuracy (43.22), Time (34.61), collaboration (14.15), and simplicity (8.02).
6.1.6.7 Observations

1. Considering all the criteria together, the relative weighting technique was ranked highest by both teams. The MoScoW technique was ranked in the second position by Team 1 and third position by Team 2. The 100-Dollar technique was ranked in the bottom position by Team 1 and in the second from the bottom position by Team 2.

2. Both teams considered accuracy their most important factor. In Team 1 simplicity, and in Team 2 time, were considered to be the second highest important criteria.

3. If we rank the prioritization techniques after evaluating it with each criterion individually, we can see in Team 1 that the MoScoW technique is ranked the highest in terms of simplicity and time criteria. Relative weighting is ranked the highest in terms of accuracy and collaboration criteria. See tables 6.30 and 6.31.
4. Results related to Team 2 are slightly different: the Top-Ten technique ranked the highest in terms of simplicity and time criteria, see tables 9 and 10. Relative weighting ranked the highest in terms of accuracy and collaboration criteria.

5. The two teams found themselves constrained by time and failed to implement their AHP results. Team 1 and Team 2 decided not to use the relative weighting tool due to its lengthy implementation. Therefore, both teams used the tool ranked the highest on each table. Team 1 used the MoScoW while Team 2 used the Top-Ten tool. That indicates there was either a problem in using the AHP with this practice, or a problem in the evaluation process. However, both teams clarified that their reason for not using the relative weighting was due to the time constraints.

<p>| Table 6.30 Prioritization technique based on each criterion individually in Team 1 |
|-----------------------------------|----------------|----------------|----------------|----------------|----------------|</p>
<table>
<thead>
<tr>
<th>Technique</th>
<th>Simplicity</th>
<th>Technique</th>
<th>Time</th>
<th>Technique</th>
<th>Accuracy</th>
<th>Technique</th>
<th>Collaboration</th>
</tr>
</thead>
<tbody>
<tr>
<td>MoScoW</td>
<td>36.35 %</td>
<td>MoScoW</td>
<td>32.67 %</td>
<td>Relative Weighting</td>
<td>39.07 %</td>
<td>Relative Weighting</td>
<td>27.76 %</td>
</tr>
<tr>
<td>Kano</td>
<td>21.09 %</td>
<td>Top-Ten</td>
<td>26.12 %</td>
<td>Them Screening</td>
<td>24.75 %</td>
<td>Them Screening</td>
<td>22.18 %</td>
</tr>
<tr>
<td>Top-Ten</td>
<td>20.65 %</td>
<td>Kano</td>
<td>15.44 %</td>
<td>Kano</td>
<td>16.54 %</td>
<td>MoScoW</td>
<td>21.09 %</td>
</tr>
<tr>
<td>100-Dollar</td>
<td>13.06 %</td>
<td>100-Dollar</td>
<td>15.35 %</td>
<td>MoScoW</td>
<td>8.59 %</td>
<td>Top-Ten</td>
<td>11.38 %</td>
</tr>
<tr>
<td>Them Screening</td>
<td>4.86 %</td>
<td>Them Screening</td>
<td>7.15 %</td>
<td>100-Dollar</td>
<td>5.74 %</td>
<td>Kano</td>
<td>10.18 %</td>
</tr>
<tr>
<td>Relative Weighting</td>
<td>3.99 %</td>
<td>Relative Weighting</td>
<td>3.27 %</td>
<td>Top-Ten</td>
<td>5.30 %</td>
<td>100-Dollar</td>
<td>7.40 %</td>
</tr>
</tbody>
</table>
Table 6.31 Prioritization technique based on each criterion individually in Team 2

<table>
<thead>
<tr>
<th>Technique</th>
<th>Simplicity</th>
<th>Technique</th>
<th>Time</th>
<th>Technique</th>
<th>Accuracy</th>
<th>Technique</th>
<th>Collaboration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top-Ten</td>
<td>31.10 %</td>
<td>Top-Ten</td>
<td>34.95 %</td>
<td>Relative Weighting</td>
<td>41.31 %</td>
<td>Relative Weighting</td>
<td>27.76 %</td>
</tr>
<tr>
<td>MoScoW</td>
<td>25.57 %</td>
<td>100-Dollar</td>
<td>20.99 %</td>
<td>Them Screening</td>
<td>22.03 %</td>
<td>Them Screening</td>
<td>22.18 %</td>
</tr>
<tr>
<td>100-Dollar</td>
<td>21.19 %</td>
<td>Kano</td>
<td>15.98 %</td>
<td>MoScoW</td>
<td>17.12 %</td>
<td>MoScoW</td>
<td>21.09 %</td>
</tr>
<tr>
<td>Kano</td>
<td>10.38 %</td>
<td>MoScoW</td>
<td>14.77 %</td>
<td>Kano</td>
<td>9.99 %</td>
<td>100-Dollar</td>
<td>11.38 %</td>
</tr>
<tr>
<td>Them Screening</td>
<td>7.57 %</td>
<td>Them Screening</td>
<td>8.89 %</td>
<td>Top-Ten</td>
<td>5.25 %</td>
<td>Kano</td>
<td>10.18 %</td>
</tr>
<tr>
<td>Relative Weighting</td>
<td>4.19 %</td>
<td>Relative Weighting</td>
<td>4.43 %</td>
<td>100-Dollar</td>
<td>4.30 %</td>
<td>Top-Ten</td>
<td>7.40 %</td>
</tr>
</tbody>
</table>
6.2 Simple Design

This section describes two proposed uses of AHP in the XP simple design practice based on certain criteria. In this study, the ideas of simple design and its tools are introduced. This section shows AHP-design hierarchy structures that include both the proposed criteria and the design tools/CRC cards as alternatives for making decisions or prioritization. The educational case studies finding and results are presented and discussed.

6.2.1 Introduction

Systems design in XP is developed incrementally. There is no need to support the next iteration when you are working on features for the current iteration. XP design meets the rules of simplicity laid down by Beck [11] which are: the application passes all the tests, demonstrates communication among developers, has no duplicate coding, and has a minimum number of classes and methods). Shalloway and Trott [183] proposed some alternative rules: it passes all the tests, contains no duplication, possesses high internal cohesion, and has loose coupling.

6.2.2 Uses of AHP in simple design

The AHP model can assist the XP development team in selecting the most beneficial design tools according to certain criteria. Another use for the AHP in the simple design practice would be to rank the CRC cards in the event a team decides to use them. Both AHP uses have been applied in the two educational cases and results were collected.
6.2.3 Firstly: Selecting the simple design tools

The first use of AHP is in selecting the best simple design tool for the XP team. The design tools used, as well as important points to consider in the process, will be discussed as follows.

Simple Design Tools:

Having a simple design makes it easier to avoid the problems created by more complex designs. However, for larger projects, more intricate designs are often considered necessary [184]. In reality there are a variety of XP design practices used by projects ranging from high-level architectural practices to low level design. As result, the XP developers have a set of design tools to choose from as follow:

1. Unified Modeling Language UML is a very powerful modeling tool used by software engineers and developers to create architectures and diagrams, which represents different perspectives from several systems. Usually, the UML is associated with a Big Design Up Front (BDUF) mentality, which might conflict with the simplicity in design.

   Nevertheless, Ambler [185] considers the question “how do you use UML with XP?”, which is the wrong question to ask in this case. He suggests that the right question should be “how do we model effectively on an XP project?” He supports using the UML minimally, and only when it is appropriate.

2. Class Responsibility Collaboration (CRC cards) is another tool used within simple design. It is used as a quick and flexible technique to discover the class of the object, its members, and the relationships within the design of the object ordinated software [186]. “It is a collection of standard index cards that have been divided
into three sections: a) class represents a collection of similar objects, b) a responsibility is something that a class knows or does, and c) collaborators are another class within the interactive spectrum that works to fulfill its responsibility” [187].

3. Whiteboard or web based white boarding: Some XP teams prefer to use this method as they see it a very simple tool. Dean Morrow (Agile Consultant) says, “I prefer to use whiteboards to create concepts, and pen & paper if none are available. We often need to collaborate across geographically dispersed teams and clients working in different time zones. A whiteboard solution seems like the most natural fit. A pen-based system would be even better (as it is not limited in size like real paper, and easier to erase). Primarily, it needs to be quick, easy, and inexpensive”[188]. Azizyan et al. [189] presented a survey conducted with 121 representatives spread over 120 companies to see the most recommended tools in Agile. The overall results showed that 26 % of the responses preferred physical walls and paper. Another interesting result that the survey showed was that of the surveyed teams, 61 % used tangible tools to brainstorm.

4. A Combination of CRC cards and UML: Borstler [190] combined some modified elements from UML’s object and collaboration diagram and developed role-play diagrams (RPDs) to be used with CRC cards. The authors found that this supports “objective thinking” and helps students understand the object-oriented programs. However, the author admitted that the RPDs are closer to Booch’s object diagram [191] than any of the current UML diagrams. “In my opinion the UML is not sufficient for the needs of business application development... UML, explicitly
includes non-UML artifacts such as business rules, data models, and user interface flow diagrams… So trying to generate a system only from UML models at the present moment simply won’t suffice”[192].

6.2.3.1 Proposed criteria for selecting the design tools

To accomplish this goal, the AHP evaluation steps explained in [31] were applied. Students were asked to evaluate the design tools based on the following criteria:

- **Simplicity**: Which design tool is the simplest to apply and simplest to understand by the team members?
- **Communication**: Which design tool aids communication among the developers?
- **Documentation**: Which design tool creates the most useful documentation?
- **Portability**: Which design tool is the easiest to transport?

6.2.3.2 AHP-simple design structure

The AHP-Simple Design Structure is composed of three levels. The primary level is the objective of selecting the design tool; the secondary level is the process of establishing the criteria of simplicity, communication, documentation and portability; and the tertiary level is composed of the working alternatives foul, CRC cards, a combination of (CRC cards and few UML diagrams), and the whiteboard. Figure 6.22 illustrates the AHP structure for the problem.
6.2.3.3 Pairwise comparison process for the design tools decision

All the students had to use every design tool during the project to see the advantages and disadvantages of each tool. They were then required to evaluate these tools based on certain criteria. For this purpose, sheets of paper with appropriate AHP tables were handed to the all students in order to save time and facilitate the process of the comparison.

The students first conferred to compare the criteria among each other, using the Saaty scale from 1-9. The participants asked the following questions:

- Which was more important: simplicity or communication and by how much?
- Which was more important: simplicity or documentation and by how much?
- Which was more important: simplicity or portability and by how much?
- Which was more important: communication or documentation and by how much?
- Which was more important: communication or portability and by how much?
- Which was more important: documentation or portability and by how much?
After finishing the comparisons based on the criteria, the students had to evaluate each of the simple design tools. As an example: In term of the simplicity, which was simpler- the UML or the CRC cards. How much simpler was it?

Similarly, all the following comparisons conducted were based on these criteria: (UML \( \times \) CRC cards), (UML \( \times \) Combination Methods), (UML \( \times \) whiteboard), (CRC cards \( \times \) Combination Methods), (CRC cards \( \times \) whiteboard), (Combination Methods \( \times \) whiteboard).

The same questions and comparisons were repeated until all the design tools were evaluated.

**6.2.3.4 AHP evaluation results for the design tools**

Team 1 ranked the design tools as follows. First: whiteboard (34.77); Second: combined methods (28.61); Third: CRC cards (20.55); Fourth: UML (16.08). Table 6.32 summarizes the results.

Figure 6.23 shows the importance of each criterion for Team 1: communication (37.53), simplicity (36.90), documentation (15.88), and portability (9.69).

<table>
<thead>
<tr>
<th>Table 6.32 Design tools ranking for Team 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Design Tools</strong></td>
</tr>
<tr>
<td>Whiteboard</td>
</tr>
<tr>
<td>Combined Methods</td>
</tr>
<tr>
<td>CRC Cards</td>
</tr>
<tr>
<td>UML</td>
</tr>
</tbody>
</table>
Team 2 ranked the design tools as follows. First: whiteboard (34.60); Second: CRC cards (30.79); Third: combined methods (17.31); Fourth: UML (17.30). Table 6.33 summarizes the results.

Figure 6.24 shows the importance of each criterion for Team 2: communication (40.71), simplicity (24.73), documentation (22.57), and portability (12.00).

<table>
<thead>
<tr>
<th>Design Tools</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whiteboard</td>
<td>34.60%</td>
</tr>
<tr>
<td>CRC Cards</td>
<td>30.79%</td>
</tr>
<tr>
<td>Combined Methods</td>
<td>17.31%</td>
</tr>
<tr>
<td>UML</td>
<td>17.30%</td>
</tr>
</tbody>
</table>

Figure 6.24 Importance of the criteria for Team 2 in the design tools selection.

6.2.3.5 Observations

Taking all the criteria into consideration, both teams ranked the whiteboard tool the highest. Team 1 ranked the combination between CRC cards and UML second, while Team 2 ranked the CRC cards alone as their second choice.

1. UML was ranked in the last position by both teams.

2. Both teams considered communication to be the most important criteria. Simplicity came in as the second important criteria for both teams.
3. If we rank the simple design tools by considering each criterion individually, we can see that both teams ranked the whiteboard as the highest with regards to simplicity and communication criteria. See tables 6.35 and 6.35.

4. Team 1 ranked the combined method the highest with regards to documentation and UML the highest with regards to in portability. See table 6.34.

5. Team 2 ranked the CRC cards the highest with regards to both documentation and portability. See table 6.35.

6. The two teams found that the AHP created an environment favoring discussion and collaboration when they started evaluating each tool.

| Table 6.34 Design tools based on each criterion individually in Team 1 |
|------------------------|------------------|------------------|------------------|------------------|------------------|
| Design Tools           | Simplicity       | Design Tools     | Communication    | Design Tools     | Documentation    | Design Tools     | Portability     |
| Whiteboard             | 60.86%           | Whiteboard       | 34.75%           | Combined Methods | 47.88%           | UML              | 32.75%          |
| CRC Cards              | 20.4%            | Combined Methods | 33.74%           | CRC Cards        | 22.79%           | CRC Cards        | 28.65%          |
| UML                    | 9.85%            | CRC Cards        | 17.92%           | UML              | 20.68%           | Combined Methods | 27.91%          |
| Combined Methods       | 9.25%            | UML              | 13.59%           | Whiteboard       | 8.66%            | Whiteboard       | 10.68%          |

| Table 6.35 Design tools based on each criterion individually in Team 2 |
|------------------------|------------------|------------------|------------------|------------------|------------------|
| Design Tools           | Simplicity       | Design Tools     | Communication    | Design Tools     | Documentation    | Design Tools     | Portability     |
| Whiteboard             | 45.84%           | Whiteboard       | 48.71%           | CRC Cards        | 39.81%           | CRC Cards        | 34.33%          |
| CRC Cards              | 28.41%           | CRC Cards        | 24.72%           | UML              | 29.96%           | UML              | 31.92%          |
| Combined Methods       | 14.00%           | Combined Methods | 20.39%           | Combined Methods | 18.89%           | Whiteboard       | 21.29%          |
| UML                    | 11.76%           | UML              | 6.18%            | Whiteboard       | 11.34%           | Combined Methods | 12.47%          |
6.2.4 Secondly: Ranking the CRC cards

The second use of AHP is to prioritize CRC cards as a common simple design tool. The CRC cards and the proposed criteria will be discussed as follows.

6.2.4.1 CRC cards overview

CRC cards were originally presented as a teaching tool for fostering object-orientated thinking among new programmers (Beck & Cunningham, 1989). Later, the CRC cards were used as tool to teach design (Biddle et al. 2002; Borstler et al. 2002)[193].

Felici [194] found certain advantages of using CRC cards in library software design. These included the facts that (1) they were good and early measure of the quality of the system design; (2) they were flexible enough to be used for recording changes during the validation process.

It has already been used successfully in XP design. “In the early days of describing XP here, the XP people were at pains to distinguish their practices from CRC. It was even suggested that XP could be done without CRC at all. Of course no one ever explained how […] so is it now fair to say that CRC is central to XP, or is that still going too far?” [195].

As stated in the wiki, “Using a small cards keeps the complexity of the design at minimum. It focuses the designer on the essentials of the class and prevents him from getting into its details and inner working at a time when such details are probably counter-productive” [196].

Alexander [197] discussed several concepts related to the CRC cards such as: the process of engaging in textual analysis, finding candidate classes, grouping the classes, and
identifying the classes. He also numbered the benefits of using the CRC cards as follows: cheap, simple, portable, intuitive, readily available, amenable to group use.

Ambler [198] summarized the steps for creating the CRC models in four steps: finding the classes, orienting the responsibility, defining the collaborators, and moving the cards around.

6.2.4.2 Proposed criteria for ranking the CRC cards

The quality of the CRC cards can be an indicator of the quality of the software design. For example, classes associated with many responsibilities will influence the coherence of the system and require more code refactoring. In contrast, a small number of responsibilities can be easier to implement and to reuse [199]. Also, the number of collaborators affects some of the code quality attributes, such as the cohesion and coupling among classes. Many collaborators can result in the coupling of the system being too high, or it can mean that the responsibilities between the classes were determined erroneously [197][199]. As result, prioritizing the CRC cards can be a way of measuring the quality of the software and ensuring the simplicity of the design.

To accomplish this goal, students were asked to prioritize the CRC cards based on the following three criteria:

- **Responsibility**: which class responsibilities affect the system more?
- **Collaboration**: which classes have the strongest connection to other classes?
- **Stability**: which class is the most stable and unchangeable?
6.2.4.3 CRC cards for the educational case studies

Each team had different CRC cards, but both followed the same procedure for evaluating them. The following sections will show the CRC cards for each team along with the AHP structures.

**CRC cards created by Team 1:**

Team 1 created four CRC cards for iteration 1 as can be seen in Tables 6.36, 6.37, 6.38, and 6.39.

### Table 6.36 Team 1: CRC1 (Class: Issue)

<table>
<thead>
<tr>
<th>Responsibilities</th>
<th>Collaborators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knows Error Message</td>
<td></td>
</tr>
<tr>
<td>Knows ID</td>
<td></td>
</tr>
<tr>
<td>Knows OwnerCompanyID</td>
<td></td>
</tr>
<tr>
<td>Knows PartnerCompanyID</td>
<td></td>
</tr>
<tr>
<td>Knows Description</td>
<td></td>
</tr>
<tr>
<td>Knows StatusID</td>
<td></td>
</tr>
<tr>
<td>Knows CategoryID</td>
<td></td>
</tr>
<tr>
<td>Knows PriorityID</td>
<td></td>
</tr>
<tr>
<td>Knows AuthoringDate</td>
<td></td>
</tr>
<tr>
<td>Knows AuthoringName</td>
<td></td>
</tr>
</tbody>
</table>

### Table 6.37 Team 1: CRC2 (Class: Assignment)

<table>
<thead>
<tr>
<th>Responsibilities</th>
<th>Collaborators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knows Error Message</td>
<td></td>
</tr>
<tr>
<td>Knows ID</td>
<td></td>
</tr>
<tr>
<td>Knows Issue ID</td>
<td>Issue</td>
</tr>
<tr>
<td>Knows Description</td>
<td></td>
</tr>
<tr>
<td>Knows PlannedStartDate</td>
<td></td>
</tr>
<tr>
<td>Knows PlannedEndDate</td>
<td></td>
</tr>
<tr>
<td>Knows ActualEndDate</td>
<td></td>
</tr>
<tr>
<td>Knows CompletionLevel</td>
<td></td>
</tr>
<tr>
<td>Knows PreviousCompletionLevel</td>
<td></td>
</tr>
<tr>
<td>Knows AuthoringName</td>
<td></td>
</tr>
<tr>
<td>Knows AuthoringDate</td>
<td></td>
</tr>
</tbody>
</table>

### Table 6.38 Team 1: CRC3 (Class: Job)

<table>
<thead>
<tr>
<th>Responsibilities</th>
<th>Collaborators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knows Error Message</td>
<td></td>
</tr>
<tr>
<td>Knows ID</td>
<td></td>
</tr>
<tr>
<td>Knows AssignmentID</td>
<td>Assignment</td>
</tr>
<tr>
<td>Knows JobType</td>
<td></td>
</tr>
<tr>
<td>Knows Description</td>
<td></td>
</tr>
<tr>
<td>Knows PlannedStartDate</td>
<td></td>
</tr>
</tbody>
</table>

### Table 6.39 Team 1: CRC4 (Class: IssuePriority)

<table>
<thead>
<tr>
<th>Responsibilities</th>
<th>Collaborators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knows Error Message</td>
<td></td>
</tr>
<tr>
<td>Knows ID</td>
<td></td>
</tr>
<tr>
<td>Knows Issue Priority</td>
<td>Issue</td>
</tr>
</tbody>
</table>

6.2.4.4 AHP-CRC cards for Team 1

The first step in the analytic hierarchy process is to structure the problem as a hierarchy that is composed of three levels. The primary level is the main objective of ranking the CRC cards; the secondary level is the responsibility of ensuring that the criteria of collaboration, and stability are met; and the tertiary level is composed of the alternatives such as CRC1 (Class: Issue), CRC2 (Class: Assignment), CRC3 (Class: Job), CRC4 (Class: Issue Priority). Figure 6.25 illustrates the AHP structure for the problem.

![Figure 6.25 AHP Structure for ranking the CRC cards for Team 1](image)

6.2.4.5 CRC cards pairwise comparison process

Similarly, sheets of paper with appropriate AHP tables were handed to the all participants in order to save time and facilitate the process of the comparison.

The participants first compared the criteria among each other using the Saaty scale from 1-9. The participants asked questions as follow:

- Which was more important: Responsibility or collaboration and by how much?
- Which was more important: Responsibility or stability and by how much?
- Which was more important: collaboration or stability and by how much?
After finishing the criteria comparisons, the participants had to evaluate all the CRC cards based on each criterion.

### 6.2.4.6 Team 1: AHP evaluation results for the CRC cards

Each student individually evaluated the CRC cards according to the criteria mentioned earlier. Expert Choice software [37] was used to calculate the aggregation results for the entire team.

Figure 6.26 shows the importance of each criterion: stability (56.90), responsibility (26.17), and collaboration (16.94).

The ranking for the CRC cards based on all the criteria (responsibilities, collaboration, and stability) is summarized as follows: First: CRC4 (32.9), Second: CRC2 (27.25), Third: CRC1 (26.81), and Forth: CRC 3 (13.04). Table 6.40 summarizes the results.

<table>
<thead>
<tr>
<th>CRC Cards</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRC4</td>
<td>32.9%</td>
</tr>
<tr>
<td>CRC2</td>
<td>27.25%</td>
</tr>
<tr>
<td>CRC1</td>
<td>26.81%</td>
</tr>
<tr>
<td>CRC3</td>
<td>13.04%</td>
</tr>
</tbody>
</table>

Table 6.40 CRC cards ranking for Team 1

![Figure 6.26 Importance of the criteria for Team 1 in the CRC cards](image)
Table 6.41 shows the order of the CRC cards for Team 1 based on each criterion individually

<table>
<thead>
<tr>
<th>CRC Cards</th>
<th>Responsibility</th>
<th>CRC Cards</th>
<th>Collaboration</th>
<th>CRC Cards</th>
<th>Stability</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRC2</td>
<td>46.79%</td>
<td>CRC2</td>
<td>38.039%</td>
<td>CRC4</td>
<td>51.13%</td>
</tr>
<tr>
<td>CRC1</td>
<td>33.35%</td>
<td>CRC3</td>
<td>26.96%</td>
<td>CRC1</td>
<td>28.14%</td>
</tr>
<tr>
<td>CRC3</td>
<td>12.75%</td>
<td>CRC4</td>
<td>19.74%</td>
<td>CRC2</td>
<td>13.11%</td>
</tr>
<tr>
<td>CRC4</td>
<td>7.11%</td>
<td>CRC1</td>
<td>15.27%</td>
<td>CRC3</td>
<td>7.62%</td>
</tr>
</tbody>
</table>

6.2.4.7 Observations

1. The CRC4 card, which is the Issue Priority Class, is the most important card to Team 1, taking all the criteria into consideration, even though it has fewer responsibilities and only one collaborator. Also, the CRC4 card was ranked in the highest position when only the stability of the class was considered. If we look at the order of the criteria ranking, stability scored 51.13%, which is higher than the responsibility and collaboration criteria. The high score of stability is one possible reason for pushing the order of the CRC4 cards to the top. Additionally, the CRC4 cards ranked in the last position with regards to responsibility (7.11%) and one before the last position in the collaboration category (19.74%).

2. The CRC2 card, which is the Assignment Class, ranks second in the order of importance. Considering each criterion individually, the CRC2 cards scored the highest score in two categories: responsibility (46.79%) and collaboration (38.03%). However, CRC2 cards have 11 responsibilities and only 1 collaborator.
3. The CRC3 card, which is the Job Class, is the least important card (13.04%) to the team, taking into consideration all the categories. CRC3 cards ranked the lowest based on the stability criterion (7.62%) and one before the last based on the responsibility criterion (12.75%).

4. The CRC1 card, which is the Issue Class, ranked one before the last position taking all the criteria into consideration. CRC1 cards ranked in the last position based on the collaboration criteria (15.27%). Even though the CRC1 card came in second in terms of the stability criterion (28.14%), which is more than the CRC2 card (13.11%), the CRC2 card is in the higher position considering all the criteria.

5. Knowledge of the number of collaborators and responsibilities is not enough to evaluate the importance of the CRC cards. For example, each of the CRC cards 2, 3, and 4, has one collaborator, but they are not all equally important. Also, the CRC1 card has no collaborator, but this does not mean that it is not important. On the other hand, all the CRC cards have a different number of responsibilities: The CRC1 card has 10 responsibilities, the CRC2 card has 11 responsibilities, the CRC3 card has 6 responsibilities, and the CRC4 card has 3 responsibilities. They all have different impacts on the software design. As a result, they have different ranks in terms of importance. In order to work towards achieving a good and simple design, AHP is the best tool to rank these cards.

6. CRC cards allow all the development team members to contribute equally to the design activity. This fosters discussion and collaboration.
CRC cards created by Team 2:

Team 2 created 4 CRC cards for iteration 1 as can be seen in Tables 6.42, 6.43, 6.44, and 6.45.

Table 6.42 Team 2: CRC1 (Class: Authentication)

<table>
<thead>
<tr>
<th>Responsibilities</th>
<th>Collaborators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knows User Name</td>
<td>AdminPanel</td>
</tr>
<tr>
<td>Knows Password</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.43 Team 2: CRC2 (Class: AdminPanel)

<table>
<thead>
<tr>
<th>Responsibilities</th>
<th>Collaborators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create User</td>
<td>EditUser</td>
</tr>
</tbody>
</table>

Table 6.44 Team 2: CRC3 (Class: EditUser)

<table>
<thead>
<tr>
<th>Responsibilities</th>
<th>Collaborators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knows Company User</td>
<td>UserForm</td>
</tr>
<tr>
<td>Knows Client User</td>
<td></td>
</tr>
<tr>
<td>Knows Vender</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.45 Team 2: CRC4 (Class: UserForm)

<table>
<thead>
<tr>
<th>Responsibilities</th>
<th>Collaborators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knows User Name</td>
<td></td>
</tr>
<tr>
<td>Knows Password</td>
<td></td>
</tr>
<tr>
<td>Knows First Name</td>
<td></td>
</tr>
<tr>
<td>Knows Last Name</td>
<td></td>
</tr>
<tr>
<td>Knows Role</td>
<td></td>
</tr>
<tr>
<td>Knows Email</td>
<td></td>
</tr>
<tr>
<td>Knows Information</td>
<td></td>
</tr>
<tr>
<td>Deletes User</td>
<td></td>
</tr>
<tr>
<td>Knows Edits User</td>
<td></td>
</tr>
</tbody>
</table>

6.2.4.8 AHP-CRC cards for Team 2

The first step in the analytic hierarchy process is to structure the problem as a hierarchy that is composed of three levels. The primary level is the main objective of ranking the CRC cards; the second level is the meeting the criteria of responsibility, collaboration, and stability; and the third level is composed of the alternatives CRC1 (Class: Authentication), CRC2 (Class: AdminPanel), CRC3 (Class: EditUser), CRC4 (Class: UserForm). Figure 6.27 illustrates the AHP structure for the problem.
6.2.4.9 Team 2: AHP evaluation results for the CRC cards

Figure 6.28 shows the importance of each criterion: collaboration (36.83), Responsibility (33.21), and Stability (29.95). The ranking for the CRC cards based on all the criteria (responsibilities, collaboration, and stability) is summarized as follows: First: CRC4 (41.02), Second: CRC3 (29.34), Third: CRC1 (71.91), and Forth: CRC1 (11.74). Table 6.46 summarizes the results.

<table>
<thead>
<tr>
<th>CRC Cards</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRC4</td>
<td>41.02%</td>
</tr>
<tr>
<td>CRC3</td>
<td>29.34%</td>
</tr>
<tr>
<td>CRC2</td>
<td>17.91%</td>
</tr>
<tr>
<td>CRC1</td>
<td>11.74%</td>
</tr>
</tbody>
</table>

Figure 6.27 AHP Structure for ranking the CRC cards for Team 2

Figure 6.28 Importance of the criteria for Team 2 in the CRC cards.
Table 6.47 shows the order of the CRC cards for Team 2 based on each criterion individually.

Table 6.47 CRC cards based on each criterion individually in Team 2

<table>
<thead>
<tr>
<th>CRC Cards</th>
<th>Responsibility</th>
<th>CRC Cards</th>
<th>Collaboration</th>
<th>CRC Cards</th>
<th>Stability</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRC4</td>
<td>56.48%</td>
<td>CRC3</td>
<td>38.80%</td>
<td>CRC4</td>
<td>42.23%</td>
</tr>
<tr>
<td>CRC3</td>
<td>23.23%</td>
<td>CRC4</td>
<td>30.53%</td>
<td>CRC2</td>
<td>22.82%</td>
</tr>
<tr>
<td>CRC1</td>
<td>11.43%</td>
<td>CRC2</td>
<td>19.83%</td>
<td>CRC3</td>
<td>21.74%</td>
</tr>
<tr>
<td>CRC2</td>
<td>8.87%</td>
<td>CRC1</td>
<td>10.83%</td>
<td>CRC1</td>
<td>13.21%</td>
</tr>
</tbody>
</table>

6.2.4.10 Observations

1. The CRC4 card, which is the UserForm Class, is the most important card for Team 2, taking all the criteria into consideration. It has nine responsibilities and only one collaborator. When considering each criterion individually, the CRC4 card ranked in the highest position in term of responsibility and stability criteria. Also, it ranked in the second position with regards to collaboration.

2. If we look at the order of the importance of these criteria, collaboration scored 36.38%, which is higher than the responsibility (33.21) and stability criteria (29.95). Considering the individuality here, The CRC3 ranked the highest with regards to collaboration, but that did not prevent it from being ranked ahead of the CRC4 in the aggregated result. One reason for this is because the CRC4 ranked higher in the two other criteria. Also, the differences between the three criteria in term of their importance are not high enough to alter the results for CRC3.

3. The CRC3 card, which is the EditUser Class, is the second most important card, when considering all the criteria together. Also, considering each criterion
individually, the CRC3 cards scored the highest with regards to collaboration (36.38) and second with regards to responsibility. However, CRC3 card had 3 responsibilities and only 1 collaborator.

4. Team 2 regarded the CRC1 card, Authentication, as the least important card (11.47). CRC1 cards ranked in the lowest position in two criteria: collaboration (10.83) and stability (13.21). Also, it ranked one below the bottom card with regards to responsibility (11.43).
6.3 Pair Programming

This chapter describes two proposed uses of AHP in pair programming based on several criteria. An introduction of pair programming and current research in the field is presented as well. This chapter shows AHP-Pair programming hierarchy structures that include the criteria and proposed alternatives for ranking. The results and findings of educational and industrial case studies are presented and discussed.

6.3.1 Introduction

Pair programming in XP means: two programmers work together on one machine to do the same task. One of them is responsible for typing the code (the driver), and the second is responsible for watching and reviewing the problem currently being worked on (the navigator) [200]. XP programmers can achieve numerous benefits from using pair programming such as: code with fewer defects, improved design quality, accelerated problem solving, timely delivery, fewer distractions, higher productivity, ensured satisfaction, better team communication, store code ownership, shared knowledge amongst team members, greater self discipline, quickly provided feedback, more mistakes caught, and better design and shorter code [201].

Williams et al. [202] conducted a study that ran with advanced undergraduate students at the University of Utah. They aimed to prove through qualitative and quantitative evidence that pair programming produces higher quality software in less time, in addition to more confident programmers.

Through a case study implemented by postgraduate students in [203], the authors explored the factors that may affect the success of pair programming, and provided answers to the question: Why is pair programming sometimes ineffective? They found
that individual skill level, changing roles, and assigned tasks are the main factors that affect the success of pair programming.

Hahn et al. [204] provided an assessment strategy to evaluate individual programming abilities during pair programming situations.

Tomayko [205] has proved that when programmers work in pairs, they made less error than the individual programming situations.

VanDeGrift [206] found out that pair programming increases the programmers’ performance and confidence. Sometimes it decreases the programmers’ frustration levels. In addition, pair programming could be a promising way to teach programming and improve the programmers’ skills.

Katira et al. [207] conducted a study involving 361 software engineering students at North Carolina State University to investigate the compatibility of pairs in pair programming. They found that students are compatible with partners whom they perceive are of a similar skill. They consider midterm grades and GPA to be indicators of skill. The authors also found that pairs with mixed genders are less likely to be compatible. They stated, “A collaborative style of programming seems to appeal more to female and minority students because of the highlighted social nature of the pair programming paradigm” [207].

Researcher Brunner found that: “The feminine take on technology looks right through the machine to its social function, while the masculine view is more likely to be focused on the machine itself. As a result, when technology is introduced as an end in itself, as in a programming class, for instance, young women are less likely to be interested than young men” [208,209].
Katira et al. [210] examined the compatibility of pairs among freshman, advanced undergraduate, and graduate students. They found that students who have a partner in the same skill level are more compatible than others. For example, graduate students work well with partners of similar actual skill level and freshmen work better with partners with the same skills.

6.3.2 Research focused in pair programming

The results of a survey-based study at the University of Wales [211] showed that students with lower self-esteem liked pair programming more than students with higher self-esteem. In addition, The National Centre for Education Statistics in [212] shows a low representation of women and minorities in computer science.

Treisman’s study titled “peer checking” was discussed by Nelson in [213], and showed that African-American success rates improved when shifting from individual work to collaborative small groups.

Salomon believed that “Knowledge is commonly socially constructed, through collaborative efforts toward shared objectives or by dialogues and challenges brought about by differences in persons’ perspectives” [214].

Dick and Zarnett [215] emphasized that several personality traits should be considered when two developers are paired to ensure an effective collaboration: effective communication, comfort working with a partner, confidence in one’s abilities and the ability to compromise.

Moreover, the initial findings indicate that pair programming produces shorter code (e.g. [216, 217]) and results in better adherence to coding standards [217].

Müller [218] reported an increase of 5% on total project costs caused by instating
pair programming.

6.3.3 Uses of AHP in pair programming

However, there is ongoing concern about finding the best matching for pairs. Many studies showed different criteria that should be considered when forming pairs in order to maximize the productivity from pair programming [219]. For example, personality type, experience, programming style, gender, and culture.

So, AHP can be used in pair programming for two purposes:

1. To select the best pair matching based on different goals like speed, learning, sharing knowledge, and code quality. In [220] there are four possible alternatives that can be investigated: (1) Expert-Expert Pairing, (2) Expert-average Pairing, (3) Expert-Novice pairing, and (4) Novice-Novice Pairing.

2. To decide if the pairs should have the same characteristics or be different. The selection based on the factors mentioned previously will be: (1) almost identical, (2) marginally different, or (3) clearly different.
6.3.4 Firstly: Selecting the best pairs

The first use of AHP is to select the best pairs based on the provided criteria. The essential criteria for pairing will be discussed in the following.

6.3.4.1 Proposed criteria for selecting the best pairs

To accomplish this goal, students were asked to evaluate pairs based on the following four main criteria:

- **Speed**: which pairs are best for speeding up the coding practice?
- **Sharing Knowledge**: which pairs are best for exchanging more knowledge?
- **Code Quality**: which pairs could help to improve the code quality?
- **Learning**: which pairs will be involved in this type of training and learning environment?

AHP will be used with different pairings that are detailed in Williams and Kessler’s book, “Pair Programming Illuminated” [221].

- **Expert-expert**: When pairing two experts, there might be an issue with big egos, but the job could be done perfectly. As Jeffries of Object Mentor says, “When the two experts get in sync, you can hear the lightning crackling. Working with a good expert partner is like gaining 40 or more IQ points” [221]. However, Lui and Chan [222] conducted an empirical study in pair programming and found that there is a greater increase in productivity when two novices are paired together (vs. working solo) than when two experts are paired together (vs. working solo).

- **Expert-average**: When an expert is paired with someone at an average level there is the possibility of raising his/her skill level. However, if the average person has
no interest in expanding his knowledge or doesn’t interact well with the expert, they might experience conflict.

- **Expert-novice:** The expert has to be willing to train the novice, which requires him/her to be more patient at times. On the other hand, the expert should welcome advice or suggestions from the novice and be able to admit mistakes if there are any.

- **Novice-novice:** “To produce production code in a relatively noncomplex area of the project, giving valuable experience to both programmers in the process [sic]” [221].

### 6.3.4.2 AHP-best pairs structure

The AHP-Best Pairs Structure is comprised of three levels: The top level is the main objective of finding the pairs; the second level is criteria speed, sharing knowledge, code quality, and learning; and the third level is the options of Expert-Expert Pairing, Expert-Average Pairing, Expert-Novice Pairing, and Novice-Novice pairing. Figure 6.29 illustrates the AHP structure for the problem.

![Figure 6.29 AHP Structure for the selecting two pairs.](image)
6.3.4.3 Pairwise comparison process for selecting pairs

All of the participants were required to evaluate these tools based on certain criteria. For this purpose, sheets of paper with appropriate AHP tables were handed to the all students in order to keep the time short and facilitate the process of comparison.

The participants first compared the criteria using the Saaty scale from 1-9. The participants were asked questions as follow:

- Which is more important: speed or sharing knowledge and by how much?
- Which is more important: speed or code quality and by how much?
- Which is more important: speed or learning and by how much?
- Which is more important: sharing knowledge or code quality and by how much?
- Which is more important: sharing knowledge or learning and by how much?
- Which is more important: code quality or learning and by how much?

After finishing the criteria comparisons, the students had to evaluate all pair alternatives based on each criterion. Example follows:

- In term of the speed, which pairs are faster and by how much is that?

Similarly, all the following options of pairs were rated based on each criterion:


The same questions and comparisons were repeated until all proposed pairs based on each criterion were evaluated.
6.3.4.4 AHP evaluation results for selecting pairs

1. Educational case studies

For Team 1, the ranking for best pairs based on all criteria, i.e. speed, sharing knowledge, code quality and learning, is summarized as follows: First: Expert-Expert (36.44); Second: Expert-Average (26.28); Third: Expert-Novice (21.74); Fourth: Novice-Novice (15.54). Table 6.48 summarizes the results.

Figure 6.30 shows the importance of each criterion as follows: code quality (56.11), sharing knowledge (23.94), learning (10.82), and speed (9.13).

<table>
<thead>
<tr>
<th>Pairs</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expert-Expert</td>
<td>36.44 %</td>
</tr>
<tr>
<td>Expert-Average</td>
<td>26.28 %</td>
</tr>
<tr>
<td>Expert-Novice</td>
<td>21.74 %</td>
</tr>
<tr>
<td>Novice-Novice</td>
<td>15.54 %</td>
</tr>
</tbody>
</table>

The ranking for the best pairs by Team 2 is summarized as follows: First: Expert-Expert (38.19); Second: Expert-Average (33.58); Third: Expert-Novice (19.59); Fourth: Novice-Novice (8.64). Table 6.49 summarizes the results.

Figure 6.31 shows the importance of each criterion as follows: code quality (61.92), learning (15.64), speed (11.22), and sharing knowledge (11.22).
6.3.4.5 Observations (educational case studies)

7. Considering all the criteria together, both teams have the same ranking; the highest rank was expert-expert, the second expert-average, then expert-novice, and finally novice-novice.

8. Both teams considered code quality to be the most important criteria. Sharing knowledge was considered the second most important criteria for Team 1, while Team 2 ranked learning as the second most important criteria.

9. If we rank the pairs considering each criterion individually, we can see that both teams ranked expert-expert the highest in terms of speed and code quality criteria, see tables 5 and 6.

10. Team 1 ranked expert-novice the highest in terms of sharing knowledge and ranked novice-novice the highest in terms of learning criteria, see table 6.50.

11. Team 2 ranked expert-average the highest in terms of sharing knowledge and learning criteria, see table 6.51.
### Table 6.50 Ranking the pairs by Team 1 based on each criterion individually

<table>
<thead>
<tr>
<th>Pairs</th>
<th>Speed</th>
<th>Pairs</th>
<th>Sharing Knowledge</th>
<th>Pairs</th>
<th>Code Quality</th>
<th>Pairs</th>
<th>Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expert-Expert</td>
<td>51.91%</td>
<td>Expert-Novice</td>
<td>35.12%</td>
<td>Expert-Expert</td>
<td>58.27%</td>
<td>Novice-Novice</td>
<td>43.95%</td>
</tr>
<tr>
<td>Expert-Average</td>
<td>30.38%</td>
<td>Novice-Novice</td>
<td>26.68%</td>
<td>Expert-Average</td>
<td>26.18%</td>
<td>Expert-Novice</td>
<td>26.54%</td>
</tr>
<tr>
<td>Novice-Novice</td>
<td>4.79%</td>
<td>Expert-Expert</td>
<td>13.84%</td>
<td>Novice-Novice</td>
<td>4.36%</td>
<td>Expert-Expert</td>
<td>14.52%</td>
</tr>
</tbody>
</table>

### Table 6.51 Ranking the pairs by Team 2 based on each criterion individually

<table>
<thead>
<tr>
<th>Pairs</th>
<th>Speed</th>
<th>Pairs</th>
<th>Sharing Knowledge</th>
<th>Pairs</th>
<th>Code Quality</th>
<th>Pairs</th>
<th>Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expert-Expert</td>
<td>39.23%</td>
<td>Expert-Average</td>
<td>32.96%</td>
<td>Expert-Expert</td>
<td>44.79%</td>
<td>Expert-Average</td>
<td>47.52%</td>
</tr>
<tr>
<td>Expert-Average</td>
<td>39.56%</td>
<td>Expert-Expert</td>
<td>31.43%</td>
<td>Expert-Average</td>
<td>29.19%</td>
<td>Expert-Novice</td>
<td>27.88%</td>
</tr>
<tr>
<td>Expert-Novice</td>
<td>15.35%</td>
<td>Expert-Novice</td>
<td>25.09%</td>
<td>Expert-Novice</td>
<td>17.13%</td>
<td>Expert-Expert</td>
<td>16.54%</td>
</tr>
<tr>
<td>Novice-Novice</td>
<td>5.87%</td>
<td>Novice-Novice</td>
<td>10.52%</td>
<td>Novice-Novice</td>
<td>8.89%</td>
<td>Novice-Novice</td>
<td>8.07%</td>
</tr>
</tbody>
</table>
2. Industrial case studies

The ranking of best pairs by company A is summarized as follows: First: Expert-Expert (39.29); Second: Expert-Average (24.96); Third: Expert-Novice (21.37); Fourth: Novice-Novice (14.38). Table 6.52 summarizes the results.

Figure 6.32 shows the importance of each criterion as follows: sharing knowledge (37.91), speed (30.30), code quality (20.04), and learning (11.75).

<table>
<thead>
<tr>
<th>Pairs</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expert-Expert</td>
<td>39.29%</td>
</tr>
<tr>
<td>Expert-Average</td>
<td>24.96%</td>
</tr>
<tr>
<td>Expert-Novice</td>
<td>21.37%</td>
</tr>
<tr>
<td>Novice-Novice</td>
<td>14.38%</td>
</tr>
</tbody>
</table>

Table 6.52 Pair ranking for company A

The ranking of best pairs by company B is summarized as follows: First: Expert-Expert (33.28); Second: Expert-Average (31.37); Third: Expert-Novice (27.96); Fourth: Novice-Novice (7.38). Table 6.53 summarizes the results.

Figure 6.33 shows the importance of each criterion as follows: code quality (34.49), sharing knowledge (25.31), speed (24.88), and learning (15.32).
The ranking of best pairs by company C is summarized as follows: First: Expert-Expert (35.97); Second: Expert-Average (28.18); Third: Expert-Novice (26.26); Fourth: Novice-Novice (9.59). Table 6.54 summarizes the results.

Figure 6.34 shows the importance of each criterion as follows: code quality (56.46), learning (16.89), sharing knowledge (16.37), and speed (9.91).
6.3.4.6 Observations (industrial case studies)

1. Considering all the criteria, the three companies had the same ranking; the highest rank was expert-expert, the second expert-average, then expert-novice, and finally novice-novice.

2. Code quality was considered the most importance criteria for companies B and C, while company A considered sharing knowledge to be the highest concern.

3. If we look at the refactoring techniques and consider each criterion individually, we can see that the three companies ranked expert-expert as the top position for both the speed and code quality criteria.

4. For the sharing knowledge criteria, companies B and C ranked expert-novice in the highest position, while company A ranked expert-expert in the top. See tables 6.56 and 6.57.

5. In term of learning, A, B, and C all ranked expert-novice in the highest position. See tables 6.55, 6.56 and 6.57.
Table 6.55 Ranking the pairs by company (A) based on each criterion individually

<table>
<thead>
<tr>
<th>Pairs</th>
<th>Speed</th>
<th>Pairs</th>
<th>Sharing Knowledge</th>
<th>Pairs</th>
<th>Code Quality</th>
<th>Pairs</th>
<th>Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expert-Expert</td>
<td>44.91 %</td>
<td>Expert-Expert</td>
<td>40.62 %</td>
<td>Expert-Expert</td>
<td>48.06 %</td>
<td>Expert-Novice</td>
<td>35.38 %</td>
</tr>
<tr>
<td>Expert-Average</td>
<td>26.43 %</td>
<td>Expert-Average</td>
<td>24.35 %</td>
<td>Expert-Average</td>
<td>29.61 %</td>
<td>Novice-Novice</td>
<td>33.62 %</td>
</tr>
<tr>
<td>Expert-Novice</td>
<td>16.52 %</td>
<td>Expert-Novice</td>
<td>23.37 %</td>
<td>Expert-Novice</td>
<td>13.58 %</td>
<td>Expert-Average</td>
<td>17.84 %</td>
</tr>
<tr>
<td>Novice-Novice</td>
<td>12.15 %</td>
<td>Novice-Novice</td>
<td>11.66 %</td>
<td>Novice-Novice</td>
<td>8.75 %</td>
<td>Expert-Expert</td>
<td>13.16 %</td>
</tr>
</tbody>
</table>

Table 6.56 Ranking the pairs by company (B) based on each criterion individually

<table>
<thead>
<tr>
<th>Pairs</th>
<th>Speed</th>
<th>Pairs</th>
<th>Sharing Knowledge</th>
<th>Pairs</th>
<th>Code Quality</th>
<th>Pairs</th>
<th>Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expert-Expert</td>
<td>42.18 %</td>
<td>Expert-Novice</td>
<td>46.02 %</td>
<td>Expert-Expert</td>
<td>52.20 %</td>
<td>Expert-Novice</td>
<td>45.77 %</td>
</tr>
<tr>
<td>Expert-Average</td>
<td>34.40 %</td>
<td>Expert-Average</td>
<td>29.35 %</td>
<td>Expert-Average</td>
<td>30.18 %</td>
<td>Expert-Average</td>
<td>31.71 %</td>
</tr>
<tr>
<td>Novice-Novice</td>
<td>6.30 %</td>
<td>Novice-Novice</td>
<td>8.15 %</td>
<td>Novice-Novice</td>
<td>4.20 %</td>
<td>Expert-Expert</td>
<td>7.87 %</td>
</tr>
</tbody>
</table>

Table 6.57 Ranking the pairs by company (C) based on each criterion individually

<table>
<thead>
<tr>
<th>Pairs</th>
<th>Speed</th>
<th>Pairs</th>
<th>Sharing Knowledge</th>
<th>Pairs</th>
<th>Code Quality</th>
<th>Pairs</th>
<th>Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expert-Expert</td>
<td>52.59 %</td>
<td>Expert-Novice</td>
<td>33.23 %</td>
<td>Expert-Expert</td>
<td>43.91 %</td>
<td>Expert-Novice</td>
<td>43.95 %</td>
</tr>
<tr>
<td>Expert-Average</td>
<td>28.41 %</td>
<td>Expert-Average</td>
<td>30.19 %</td>
<td>Expert-Average</td>
<td>27.85 %</td>
<td>Expert-Average</td>
<td>26.54 %</td>
</tr>
<tr>
<td>Novice-Novice</td>
<td>4.92 %</td>
<td>Novice-Novice</td>
<td>11.09 %</td>
<td>Novice-Novice</td>
<td>8.21 %</td>
<td>Novice-Novice</td>
<td>14.52 %</td>
</tr>
</tbody>
</table>
6.3.5 Secondly: Ranking the rules of matching two persons

The second use of AHP is to select the best rules when matching pairs. The rules of matching and the proposed criteria will be discussed as follows.

6.3.5.1 Proposed criteria for rules of matching two persons

Several researchers have identified and investigated various human factors that affect the interaction between partners. These factors include diverse issues such as:

- **Personality**: attributes such as adventurous, charismatic, creative, friendly, flexible, ambitious, open-minded, quiet, and calm. Shneiderman stated in his book *Software Psychology*: “Personality variables play a critical role in determining interaction among programmers and in the work style of individual programmers” [223].

- **Experience**: in industry or developing big projects.

- **Programming Style**: rules or guidelines followed when writing the code.

- **Gender**: male or female. Pair programming can be beneficial for women “because it addresses factors that potentially limit their participation in CS. The collaborative nature of pair programming teaches women students that software development is not the competitive, socially isolating activity that they imagined”[224].

- **Culture**: “Having pairs with different cultural backgrounds is wonderful for building trust and communication within the team. As long as there is communication, the pair can succeed” [221].
5.3.5.2 Rules of pair matching

Considering these criteria, we have three options when pairing two developers:
These characteristics should be identical, marginally different, or clearly different.
What is the best option to take? Considering each criterion individually, what will be the
decision? These questions can be answered using the AHP.

6.3.5.3 AHP-rules of pair matching structure

The AHP- Rules of Pair Matching Structure includes three levels: The top level is
the main objective dictating the rules of pair matching; the second level houses the
criteria speed, sharing knowledge, code quality and learning; and the third level is the
options Expert-Expert Pairing, Expert-Average Pairing, Expert-Novice Pairing, Novice-
Novice pairing. Figure 6.35 illustrates the AHP structure for the problem.

![Diagram of AHP structure for rules in pair matching]

Figure 6.35 AHP structure for the rules in pair matching
6.3.5.4 Pairwise comparison process for rules of matching

Sheets of paper with the appropriate AHP tables were given to all participants in order to facilitate the comparison process.

The participants compared the criteria using the Saaty scale from 1-9. The participants were asked questions as follow:

- Which is more important: personality or experience and by how much?
- Which is more important: personality or programming styles and by how much?
- Which is more important: personality or gender and by how much?
- Which is more important: personality or culture and by how much?
- Which is more important: experience or programming styles and by how much?
- Which is more important: experience or gender and by how much?
- Which is more important: experience or culture and by how much?
- Which is more important: programming styles or gender and by how much?
- Which is more important: programming styles or culture and by how much?
- Which is more important: gender or culture and by how much?

After finishing the criteria comparisons, the participants had to evaluate the matching rules based on each criterion, in the same way as shown previously with the best pairs.
6.3.5.5 AHP evaluation results for rules of matching

1. Educational case studies

For Team 1, the rankings based on all criteria, i.e. personality, experience, programming style, culture and gender, are summarized as follows. First: almost identical (46.73); Second: marginally different (38.26); Third: clearly different (15.00). Table 6.58 summarizes the results.

Figure 6.36 shows the importance of each criterion as follows: experience(40.31), personality (19.78), culture (17.64), programming style (16.47) and gender (5.79).

<table>
<thead>
<tr>
<th>Rules In Pairing</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Almost Identical</td>
<td>46.73 %</td>
</tr>
<tr>
<td>Marginally Different</td>
<td>38.26 %</td>
</tr>
<tr>
<td>Clearly Different</td>
<td>15.00 %</td>
</tr>
</tbody>
</table>

![Table 6.58 Ranking the rules of pairing for Team 1](image)

The ranking of best pairs by Team 2 is summarized as follows: First: marginally different (47.79); Second: almost identical (35.60); Third: clearly different (16.61). Table 6.59 summarizes the results.

Figure 6.37 shows the importance of each criterion as follows: experience (53.73), personality (19.22), programming style (16.68), culture (5.21), and gender (5.17).
6.3.5.6 Observations (educational case studies)

1. Based on all of the matching rules criteria, Team 1 ranked the almost identical option in the first position, while Team 2 ranked marginally different as first.

2. Neither team preferred the option of clearly different, so both teams ranked it in the last position.

3. Both teams had the same order for the importance for the matching rules criteria. Experience ranked the highest and gender ranked last.

4. If we rank the rules of matching pairs considering each criterion individually, we can see that Team 1 ranked almost identical as the highest in terms of personality, experience, programming style, culture, and gender, see table 6.60.

5. Team 2 ranked almost identical as the highest in terms of programming style and culture. Also, they ranked marginally different as the highest in terms of personality and experience criteria, and clearly different in terms of gender. See table 6.61.
Table 6.60 Rules of pairing by Team 1 based on each criterion individually

<table>
<thead>
<tr>
<th>Rules In Pairing</th>
<th>Personality</th>
<th>Rules In Pairing</th>
<th>Experience</th>
<th>Rules In Pairing</th>
<th>Programming Style</th>
<th>Rules In Pairing</th>
<th>Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>Almost Identical</td>
<td>43.34 %</td>
<td>Almost Identical</td>
<td>42.58 %</td>
<td>Almost Identical</td>
<td>53.36 %</td>
<td>Almost Identical</td>
<td>47.99 %</td>
</tr>
<tr>
<td>Marginally Different</td>
<td>42.87 %</td>
<td>Marginally Different</td>
<td>39.15 %</td>
<td>Marginally Different</td>
<td>37.14 %</td>
<td>Marginally Different</td>
<td>31.43 %</td>
</tr>
<tr>
<td>Clearly Different</td>
<td>13.79 %</td>
<td>Clearly Different</td>
<td>18.28 %</td>
<td>Clearly Different</td>
<td>9.50 %</td>
<td>Clearly Different</td>
<td>20.59 %</td>
</tr>
</tbody>
</table>

Rules In Pairing | Culture |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Almost Identical</td>
<td>57.44 %</td>
</tr>
<tr>
<td>Marginally Different</td>
<td>32.51 %</td>
</tr>
<tr>
<td>Clearly Different</td>
<td>10.05 %</td>
</tr>
</tbody>
</table>

Table 6.61 Rules of pairing by Team 2 based on each criterion individually

<table>
<thead>
<tr>
<th>Rules In Pairing</th>
<th>Personality</th>
<th>Rules In Pairing</th>
<th>Experience</th>
<th>Rules In Pairing</th>
<th>Programming Style</th>
<th>Rules In Pairing</th>
<th>Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marginally Different</td>
<td>49.88 %</td>
<td>Marginally Different</td>
<td>57.25 %</td>
<td>Almost Identical</td>
<td>68.43 %</td>
<td>Clearly Different</td>
<td>35.29 %</td>
</tr>
<tr>
<td>Almost Identical</td>
<td>35.41 %</td>
<td>Almost Identical</td>
<td>25.88 %</td>
<td>Marginally Different</td>
<td>23.77 %</td>
<td>Marginally Different</td>
<td>33.29 %</td>
</tr>
<tr>
<td>Clearly Different</td>
<td>14.70 %</td>
<td>Clearly Different</td>
<td>16.87 %</td>
<td>Clearly Different</td>
<td>7.81 %</td>
<td>Almost Identical</td>
<td>31.32 %</td>
</tr>
</tbody>
</table>

Rules In Pairing | Culture |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Almost Identical</td>
<td>54.60 %</td>
</tr>
<tr>
<td>Marginally Different</td>
<td>29.91 %</td>
</tr>
<tr>
<td>Clearly Different</td>
<td>15.49 %</td>
</tr>
</tbody>
</table>
2. **Industrial case studies**

The rankings for the rules of pair matching by company A are summarized as follows. First: almost identical (51.06); Second: marginally different (29.77); Third: clearly different (19.17). Table 6.62 summarizes the results.

Figure 6.38 shows the importance of each criterion as follows: experience (32.00), programming style (29.36), personality (20.96), culture (11.04), and gender (6.64).

<table>
<thead>
<tr>
<th>Rules In Pairing</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Almost Identical</td>
<td>51.06 %</td>
</tr>
<tr>
<td>Marginally Different</td>
<td>29.77 %</td>
</tr>
<tr>
<td>Clearly Different</td>
<td>19.17 %</td>
</tr>
</tbody>
</table>

The rankings for the rules of pair matching by company B are summarized as follows. First: almost identical (45.19); Second: marginally different (38.61); Third: clearly different (16.20). Table 6.63 summarizes the results.

Figure 6.39 shows the importance of each criterion as follows: experience (32.45), programming style (22.70), personality (21.33), culture (13.97), and gender (9.55).
The rankings for the rules of pair matching by company Care summarized as follows. First: almost identical (40.75); Second: marginally different (32.07); Third: clearly different (27.18). Table 6.64 summarizes the results.

Figure 6.40 shows the importance of each criterion as follows: experience (49.76), programming style (21.75), personality (15.76), culture (6.92), and gender (5.82).
6.3.5.7 Observations (industrial case studies)

1. Considering all the criteria, the three companies had the same ranking; the highest rank was almost identical, the second marginally different, the last was clearly different.

2. Similarly, the three companies had the same order for importance of the criteria: experience, programming style, personality, culture, and gender.

3. If we rank the rules of matching pairs considering each criterion individually, we can see that the three companies had nearly identical top rankings in all of the criteria except two: personality in company B and gender in company C. See tables 6.65, 6.66, and 6.67

4. Company A ranked marginally different at the top in personality, while company B ranked clearly different the top rank. See tables 6.65 and 6.66.
Table 6.65 Rules of pairing by company (A) based on each criterion individually

<table>
<thead>
<tr>
<th>Rules In Pairing</th>
<th>Personality</th>
<th>Rules In Pairing</th>
<th>Experience</th>
<th>Rules In Pairing</th>
<th>Programming Style</th>
<th>Rules In Pairing</th>
<th>Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>Almost Identical</td>
<td>52.8 %</td>
<td>Almost Identical</td>
<td>64.67 %</td>
<td>Almost Identical</td>
<td>44.49 %</td>
<td>Almost Identical</td>
<td>33.8 %</td>
</tr>
<tr>
<td>Marginally Different</td>
<td>33.17 %</td>
<td>Marginally Different</td>
<td>22.17 %</td>
<td>Marginally Different</td>
<td>31.1 %</td>
<td>Marginally Different</td>
<td>33.33 %</td>
</tr>
<tr>
<td>Clearly Different</td>
<td>14.03 %</td>
<td>Clearly Different</td>
<td>13.16 %</td>
<td>Clearly Different</td>
<td>24.4 %</td>
<td>Clearly Different</td>
<td>32.87 %</td>
</tr>
</tbody>
</table>

Table 6.66 Rules of pairing by company (B) based on each criterion individually

<table>
<thead>
<tr>
<th>Rules In Pairing</th>
<th>Personality</th>
<th>Rules In Pairing</th>
<th>Experience</th>
<th>Rules In Pairing</th>
<th>Programming Style</th>
<th>Rules In Pairing</th>
<th>Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marginally Different</td>
<td>41.77 %</td>
<td>Almost Identical</td>
<td>42.63 %</td>
<td>Almost Identical</td>
<td>53.57 %</td>
<td>Almost Identical</td>
<td>42.71 %</td>
</tr>
<tr>
<td>Almost Identical</td>
<td>41.51 %</td>
<td>Marginally Different</td>
<td>40.01 %</td>
<td>Marginally Different</td>
<td>35.89 %</td>
<td>Marginally Different</td>
<td>32.52 %</td>
</tr>
<tr>
<td>Clearly Different</td>
<td>16.72 %</td>
<td>Clearly Different</td>
<td>17.56 %</td>
<td>Clearly Different</td>
<td>9.63 %</td>
<td>Clearly Different</td>
<td>24.77 %</td>
</tr>
</tbody>
</table>

Table 6.67 Rules of pairing by company (C) based on each criterion individually

<table>
<thead>
<tr>
<th>Rules In Pairing</th>
<th>Personality</th>
<th>Rules In Pairing</th>
<th>Experience</th>
<th>Rules In Pairing</th>
<th>Programming Style</th>
<th>Rules In Pairing</th>
<th>Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>Almost Identical</td>
<td>41.4 %</td>
<td>Almost Identical</td>
<td>40.05 %</td>
<td>Almost Identical</td>
<td>49.07 %</td>
<td>Almost Identical</td>
<td>34.75 %</td>
</tr>
<tr>
<td>Clearly Different</td>
<td>30.68 %</td>
<td>Marginally Different</td>
<td>33.41 %</td>
<td>Marginally Different</td>
<td>30.41 %</td>
<td>Marginally Different</td>
<td>33.31 %</td>
</tr>
<tr>
<td>Marginally Different</td>
<td>27.92 %</td>
<td>Clearly Different</td>
<td>26.54 %</td>
<td>Clearly Different</td>
<td>20.52 %</td>
<td>Clearly Different</td>
<td>31.93 %</td>
</tr>
</tbody>
</table>

Rules In Pairing
- Personality
- Experience
- Programming Style
- Gender
6.4 Refactoring

This chapter describes two proposed uses of AHP in refactoring techniques based on internal and external code quality attributes. An introduction to the refactoring concept and the current research in the field is presented as well. This chapter shows AHP-refactoring hierarchy structures that include the quality attributes as criteria and the refactoring techniques as alternatives for ranking. The educational and industrial case studies’ findings and results are presented and discussed. In the end of this chapter, the validation by two companies and the researcher observations are presented.

6.4.1 Introduction

Code refactoring is the process of improving the design of the existing code by changing its internal structure without changing its external behavior [118]. It is a core activity of the XP development cycle to improve the design of the software and reduce the effort and cost of coding and testing. Most of the current studies focus on the following issues:

(1) Guidelines in the refactoring process

Mens and Tourwe [224] explained the refactoring steps in detail, which can be summarized as follows:

- Identifying the part of the software that should be refactored.
- Deciding which refactoring methods should be applied
- Applying the refactoring
- Assessing the effects of the applied refactoring methods on the code quality attributes
Kataoka et al. [225] provided a 3-step model: “identification of refactoring candidates, validation of refactoring effects, and application of refactoring”[255]. Other researcher-provided similar processes can be found in [226,227].

(2) Issues regarding refactoring tools

Maticorna and Perez [228] presented the refactoring characterization and showed how it can be used as a tool to compare different refactoring definitions, including refactoring catalogs. Also, the authors tackled various refactoring issues, such as design and languages, scope, actions, and application on scheduling, which can each be a good starting point for the builders of the refactoring tools.

Murphy-Hill et al. [229] performed an empirical study comparing four methods that were used to gather refactoring data to help in building a powerful refactoring tool. Simmonds and Mens [230] compared four software-refactoring tools: SmalltalkWorks 7.0, Eclipse, Guru, and Together ControlCenter 6.0. They provided detailed results that show the strengths and weakness of each tool.

Mahmood and Reddy [231] evaluated the usability of three different refactoring tools: IntelliJ IDEA 7.0.4, JBuilder 2008, and RefactorIT 2.7 beta. They suggested some improvements to increase the consistency of software application usability and to prevent human error caused by the manual refactoring. The refactoring tools were compared based on these eight characteristics: consistency, ease of use, errors, user control, user experience, design for the user, information processing, and goal assessment.

Murphy-Hill [232] created a model of how most of the refactoring tools work in the style of the refactoring browser. This model is summarized in three steps: identify, initiate and execute.
Brunel et al. [233] investigated the accuracy of refactoring tools by analyzing seven open-source Java systems (MegaMek, JasperReports, Antlr, Tyrant, PDFBox, Velocity, HSQLDB).

To select the most appropriate refactoring tools, Marija and Kresimir [234] evaluated seven tools: IntelliJ Idea (Java), Eclipse (C++, Java), Refactorit (C++, Java), NDepend (.NET code base), Refactor (C# VB.NET, C++, ASP.NET), Refactor (C# VB.NET, ASP.NET), and Refactoring Browser (Smalltalk). They compared these refactoring tools among each other with respect to automation and coverage, reliability, configurability, scalability and discoverability.

Roberts et al. [235] discussed the technical requirements and practical criteria for the refactoring tools. They emphasize that speed and integration are the most practical criteria. Also, the accuracy and the ability to search across the entire program are the most technical requirements.

Murphy-Hill et al. [236] analyzed four huge groups of data to explain how programmers usually practice the refactoring activity both with and without using the refactoring tools.

(3) Identification of code smells to locate possible refactoring

Sandalski et al. [237] used an intelligent assistant and a refactoring agent to analyze the refactoring architecture and assess the existing code to highlight the portions of code that needed to be refactored and to provide options for the methods.

Advani et al. [238] conducted an empirical study using open sources Java software to identify the area of complexity across the systems when refactoring was being applied.
Also, they discovered where the refactoring effort was being made. They also created a way for developers to decide how to allocate the testing effort.

Hayashi et al. [239] proposed a technique using plug-ins for Eclipse to guide the developer on how and where to do refactoring using the histories of program modification. This technique answered three main questions: where to refactor? Which suitable refactoring method should be used? When should refactoring should apply?

Bryton et al. [240] proposed a model called Binary Logistic Regression (BLR) to detect the code smell particularly Long Method objectively.

(4) The impact of refactoring on the internal and external quality attributes

It will be presented in detail in the following sections.

6.4.2 Refactoring Patterns (Techniques)

Fowler [118] assures that the refactoring helps developers to program faster, to find bugs and to improve the software design. So, he defined more than 70 different refactoring patterns and organized them into six categories: composing methods, moving features between objects, organizing data, simplifying conditional expressions, making methods calls simpler, and dealing with generalization. Each of the refactoring patterns has a specific purpose and effect over the quality attributes. However, projects can have different priorities in terms of the quality attributes. Using one or more of the refactoring methods variously improves the code and the design of the software. So, it is essential to allocate the team’s efforts to the most important quality attributes in order to maximize the value expected from the system. It is often unclear to software designers how to use
refactoring methods to improve particular quality attributes [241]. The task of selecting the refactoring patterns is time-consuming and can create a conflict between the programmers’ opinions.

Piveta et al. [242] introduced the AHP for three techniques and compared them based on the use of these patterns, particularly in terms of simplicity, reusability, and comprehensibility. These patterns would be more beneficial if we could investigate more patterns and rank them based on their influence on the code instead of their capabilities and uses.

Therefore, in this section, we focused on ranking refactoring patterns based on their effects on the code quality rather than their characteristics of uses. Also, we have chosen eight refactoring patterns from four different categories proposed by Fowler to show the importance of these refactoring techniques using AHP. The following patterns were selected:

- Extract Method, Inline Method, and Inline Temp Method from “Composing Methods” category.
- Extract Class, Inline Class, and Move Method from “Moving Features Between Objects” category.
- Rename Method from “Making Method Calls Simpler” category.
- Pull Up Method from “Dealing with Generalization” category.

### 6.4.3 Uses of AHP in refactoring

The AHP model can help the XP development team to rank the refactoring patterns based on the code quality attributes. In the following section, the AHP will
evaluate the refactoring methods based on some of the internal and external quality attributes.

### 6.4.4 Firstly: Refactoring patterns based on internal quality attributes:

Before applying the AHP to the refactoring patterns, I will highlight some of the previous studies that have investigated the impact of refactoring patterns on the internal code quality attributes as follow:

Moser et al. [243] conducted a case study to assess the refactoring impacts in an industrial environment. They applied Extract Method, Rename Method, Simplify Conditional, and Move Method/Field. They found that refactoring improved software quality attributes such as coupling, cohesion, and response for a class. In addition to that, they noted that team productivity increased.

Stroggylos and Spinellis [244] analyzed source code version control system logs from four open source software systems to detect changes that occurred after refactoring and to examine the refactoring’s impact on the software metrics process. The metrics examined include coupling, cohesion, and number of methods. Their results indicated that the refactoring methods used caused classes to become less coherent.

Elish and Alshayeb [245] classified the refactoring techniques based on their effects on the internal and external quality attributes. The following refactoring methods were chosen: Chain Constructors, Compose Method, Form Template Method, Introduce Null Object, Replace Conditional Dispatcher with Command, Replace Constructors with Creation Methods, and Unify Interface. These internal quality metrics were studied: LOCC (Lines of Code for Class), NOTC (Number of Test Cases) to represent the size of
a test suite, also DIT, FOUT, LCOM, LOC, NOC (Number of Children), NOF (Number of Fields), NOM (Number of Methods), as well as others.

Bois et al. [246] analyzed how refactoring changes the coupling and cohesion characteristics. These refactoring methods were used: Move Method, Replace Method with Method Object, Replace Data Value with Object and Extract Class. The study led to an improvement in the code quality.

Moser et al. [247] conducted a case study in agile environment in order to analyze the impact of refactoring on the internal quality metrics of source code. The quality metrics selected for this study were complexity, coupling, cohesion, number of methods per class, response of a class, depth of inheritance tree, and number of children. They used a methodology proposed in their research that led them to the conclusion that the refactoring could significantly improve the internal measures for reusability of object-oriented classes written in Java.

Ratzinger et al. [248] analyzed the history of a large industrial system during 15 months and showed how refactoring can reduce the change couplings and enhance the software evolvability.

Bois and Mens [249] proposed a formalist framework for the internal program quality such as number of methods, number of children, response for a class, cohesion, and coupling. For this purpose, the author studied Extract Method, Encapsulate Filled, and Pull Up Method.

Stroulia and Kapoor [250] investigated how the refactoring could improve the design and code quality. After applying some of the refactoring methods, like Extract Superclass and Extract Abstract Class, they found that the average LOC, the average the
number of statements, the number of methods, and the number of collaborators were decreased in the individual system classes.

Demeyer [251] analyzed whether refactoring has an impact on program performance when replacing conditional logic with polymorphism, the results were positive.

Yu et al. [252] performed a case study using a modeling framework guided by multiple soft goals to show that refactoring can be measured as the transformation on the state of program in the quality space.

Kataoka et al. [253] found that refactoring techniques such as extract method and Extract Class reduce the coupling in the code and improve the maintainability of the system.

Sahraoui et al. [254] detected a potential part of the system that needed to be refracted and improved. They used an empirical study to investigate the impact of the inheritance and coupling metrics on maintainability.

Dallal and Briand [255] proposed an automated refactoring approach to improve the software cohesion, as step to improve the program testability.

Geppert et al. [256] conducted an empirical study to explore the impact of refactoring a legacy system on changeability based on three factors: customer reported defect rates, effort, and scope of changes.

Tahvildari and Kontogiannis [257] improved design quality by providing a reengineering process model and a framework for object-oriented metrics that can be used as indicators for automatically detecting where transformations can be applied in the code.
Vasudeva and Shrivastava [258] conducted a study titled “Inventory Application” to improve the quality of the code size and complexity.

Zhao and Hayes [259] introduced a rank-based software using a measure-driven refactoring decision to support the development team’s decisions of where to apply resources when they did refactoring. They presented two case studies that examined the approach that identifies which classes and packages need to be refactored based on static measures such as code size, coupling, and complexity.

Geppert et al. [260] discussed strategies for refactoring in the legacy business communication product. They presented a case study showing the impact of refactoring on the system. They found the defect rates and change efforts decreased significantly.

Deursen and Moonen [261] discussed the relation between refactoring and test-first practice, two key activities in XP development. They focused on the refactoring methods that mainly have effects on the test code. Then, they proposed the notion of test-first refactoring using the test cases as a first step to finding the code-level refactoring.

**6.4.4.1 Proposed criteria for ranking the refactoring patterns**

To rank the refactoring patterns, it is necessary to identify the quality attributes that are more valuable to the development team or the organization. Each project can have a different set of criteria and alternative refactoring methods in order to be ranked and evaluated. In this paper, we have chosen four internal quality attributes as the core criteria for ranking the refactoring techniques:

- **Cohesion:** each system component does one thing and does it well.
- **Coupling:** the degree to which each system component relies on another component.
• Complexity: the degree of connectivity between elements of a design unit.
• Code size: most common technical sizing method is number of Lines Of Code (#LOC) per technology, number of files, functions, classes, tables, etc.

6.4.4.2 AHP-refactoring structure for the internal attributes

The first step in the analytic hierarchy process is to structure the problem as a hierarchy that includes three levels. The top level is the main objective: ranking the refactoring techniques; the second level is the criteria: complexity, cohesion, coupling, and code size; the third level is the alternatives: Extract Method, Inline Method, InlineTemp Method, Extract Class, Inline Class, Move Method, Pull Up Method, and Rename Method.

Figure 6.41 illustrates the AHP structure for the problem.

![Figure 6.41 AHP Structure for the refactoring techniques based on the internal attributes](image)

6.4.4.3 Refactoring patterns pairwise comparison process

All the participants had to apply the refactoring patterns to a real project to see the real impact on their code. Then they were required to evaluate the refactoring patterns based on certain criteria. For this purpose, sheets of paper with appropriate AHP tables
were handed to the all participants in order to save time and facilitate the process of comparison. The participants compared the criteria using the Saaty scale from 1-9. The participants were asked these questions:

- Which is more important: complexity or cohesion and by how much?
- Which is more important: complexity or coupling and by how much?
- Which is more important: complexity or code size and by how much?
- Which is more important: cohesion or coupling and by how much?
- Which is more important: cohesion or code size and by how much?
- Which is more important: coupling or code size and by how much?

After finishing the criteria comparisons, the participants had to evaluate all the refactoring techniques based on each criterion every time. Example follows:

- In term of reducing the complexity, which is more important Extract Method or Inline Method and by how much?

Similarly, all the following comparisons were conducted based on each criterion:

- (Extract Method X Inline Method), (Extract Method X Inline Temp Method), (Extract Method X Extract Class), (Extract Method X Inline Class), (Extract Method X Move Method), (Extract Method X Inline Pull Up Method) (Extract Method X Rename Method).
- (Inline Method X Inline Temp Method), (Inline Method X Extract Class), (Inline Method X Inline Class), (Inline Method X Move Method) (Inline Method X Inline Method), (Inline Method X Inline Pull Up Method), (Inline Method X Rename Method).
• (Inline Temp Method X Extract Class), (Inline Temp Method X Inline Class),
  (Inline Temp Method X Move Method), (Inline Temp Method X Pull Up Method), (Inline Temp Method X Rename Method).
• (Extract Class X Inline Class), (Extract Class X Move Method), (Extract Class X Pull Up Method), (Extract Class X Rename Method).
• (Inline Class X Move Method), (Inline Class X Pull Up Method), (Inline Class X Rename Method).
• (Move Method X Pull Up Method) (Move Method X Rename Method).
• (Pull Up Method Rename Method).

The same questions and comparisons repeated until the participants evaluated all refactoring techniques based on each criterion.

6.4.4.4 AHP evaluation results for the refactoring patterns

1. Educational case studies

For Team 1, the rankings of the refactoring techniques based on all criteria, i.e. complexity, cohesion, coupling and code size, are summarized as follows. First: Extract Class (17.61); Second: Extract Method (15.37); Third: Inline Class (15.35); Fourth: Pull Up Method (13.71); Fifth: Move Method (11.55); Sixth: Inline Temp method (10.96); Seventh: Inline Method (10.17); Eighth (5.28). Table 6.68 summarizes the results.

Figure 6.42 shows the importance of each criterion as follows: coupling (29.29), cohesion (29.25), code size (22.56), and complexity (18.90).
Team 2’s rankings of the prioritization techniques is summarized as follows:

First: Inline Class (19.31); Second: Inline Method (15.65); Third: Extract Class (13.42);
Fourth: Extract Method (13.35); Fifth: Move Method (12.19); Sixth: Inline Temp Method
(9.69); Seventh: Inline Method (9.56); Eighth: Rename Method (6.83). Table 6.69 summarizes the results.

Figure 6.43 shows the importance of each criterion as follows: coupling (33.61),
cohesion (32.37), complexity (27.10), and code size (6.93).
6.4.4.5 Observations (educational case studies)

12. Considering all the criteria, the Extract Class technique was ranked the highest by Team 1, while Team 2 ranked it in the third position. Team 2 ranked the Inline Class in the highest position while Team 1 ranked it in the third position.

13. Both Extract Class and Inline Class are categorized by Fowler as “Moving Features Between Objects.”

14. Team 1 ranked Extract Method in the second position, and Team 2 ranked the Inline Method in the second position. Both Extract Method and Inline Method are categorized by Fowler as “Composing Methods”.

15. Rename Method was ranked in the last position by the two teams.

16. Cohesion and coupling quality attributes were of the highest concern for both teams.
17. If we look at the refactoring techniques considering each criterion individually, we can see both teams ranked the Extract Method in the top position in the complexity attributes. Also, both teams ranked the Inline Class in the top position in the cohesion attributes. See tables 6.70 and 6.71.

18. For the coupling quality attribute, Team 1 ranked the Extract Method in the highest position, while Team 2 ranked the Inline Class at the top. See tables 6.70 and 6.71.

19. For the code size quality attribute, Team 1 ranked the Extract Class in the highest position, while Team 2 ranked the Inline Method at the top. See tables 6.70 and 6.71.
Table 6.70 Refactoring techniques based on each internal criterion by Team 1

<table>
<thead>
<tr>
<th>Refactoring Techniques</th>
<th>Complexity</th>
<th>Refactoring Techniques</th>
<th>Coupling</th>
<th>Refactoring Techniques</th>
<th>Chohesion</th>
<th>Refactoring Techniques</th>
<th>Code Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract Method</td>
<td>19.30 %</td>
<td>Extract Method</td>
<td>18.54 %</td>
<td>Inline Class</td>
<td>18.17 %</td>
<td>Extract Class</td>
<td>21.50 %</td>
</tr>
<tr>
<td>Extract Class</td>
<td>19.29 %</td>
<td>Extract Class</td>
<td>16.62 %</td>
<td>Pull Up Method</td>
<td>16.41 %</td>
<td>Inline Class</td>
<td>17.24 %</td>
</tr>
<tr>
<td>Inline Class</td>
<td>15.00 %</td>
<td>Pull Up Method</td>
<td>12.68 %</td>
<td>Extract Class</td>
<td>15.02 %</td>
<td>Inline Temp Method</td>
<td>13.99 %</td>
</tr>
<tr>
<td>Pull Up Method</td>
<td>14.70 %</td>
<td>Move Method</td>
<td>12.44 %</td>
<td>Inline Method</td>
<td>11.83 %</td>
<td>Extract Method</td>
<td>12.64 %</td>
</tr>
<tr>
<td>Move Method</td>
<td>13.16 %</td>
<td>Inline Class</td>
<td>11.43 %</td>
<td>Extract Method</td>
<td>11.65 %</td>
<td>Inline Method</td>
<td>11.16 %</td>
</tr>
<tr>
<td>Inline Method</td>
<td>7.76 %</td>
<td>Inline Temp Method</td>
<td>11.30 %</td>
<td>Inline Temp Method</td>
<td>11.56 %</td>
<td>Move Method</td>
<td>10.32 %</td>
</tr>
<tr>
<td>Inline Temp Method</td>
<td>7.11 %</td>
<td>Inline Method</td>
<td>9.33 %</td>
<td>Move Method</td>
<td>10.50 %</td>
<td>Pull Up Method</td>
<td>10.20 %</td>
</tr>
<tr>
<td>Rename Method</td>
<td>3.68 %</td>
<td>Rename Method</td>
<td>7.66 %</td>
<td>Rename Method</td>
<td>4.87 %</td>
<td>Rename Method</td>
<td>3.84 %</td>
</tr>
</tbody>
</table>

Table 6.71 Refactoring techniques based on each internal criterion by Team 2

<table>
<thead>
<tr>
<th>Refactoring Techniques</th>
<th>Complexity</th>
<th>Refactoring Techniques</th>
<th>Coupling</th>
<th>Refactoring Techniques</th>
<th>Chohesion</th>
<th>Refactoring Techniques</th>
<th>Code Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract Method</td>
<td>17.88 %</td>
<td>Inline Class</td>
<td>21.91 %</td>
<td>Inline Class</td>
<td>22.47 %</td>
<td>Inline Method</td>
<td>17.23 %</td>
</tr>
<tr>
<td>Extract Class</td>
<td>17.26 %</td>
<td>Inline Method</td>
<td>21.68 %</td>
<td>Inline Method</td>
<td>16.39 %</td>
<td>Inline Temp Method</td>
<td>14.89 %</td>
</tr>
<tr>
<td>Inline Class</td>
<td>14.97 %</td>
<td>Extract Method</td>
<td>14.57 %</td>
<td>Extract Class</td>
<td>15.40 %</td>
<td>Inline Class</td>
<td>14.44 %</td>
</tr>
<tr>
<td>Pull Up Method</td>
<td>13.78 %</td>
<td>Move Method</td>
<td>12.12 %</td>
<td>Move Method</td>
<td>13.47 %</td>
<td>Pull Up Method</td>
<td>13.74 %</td>
</tr>
<tr>
<td>Move Method</td>
<td>11.40 %</td>
<td>Inline Temp Method</td>
<td>10.62 %</td>
<td>Inline Temp Method</td>
<td>10.27 %</td>
<td>Extract Method</td>
<td>11.11 %</td>
</tr>
<tr>
<td>Rename Method</td>
<td>9.45 %</td>
<td>Extract Method</td>
<td>8.59 %</td>
<td>Pull Up Method</td>
<td>9.08 %</td>
<td>Move Method</td>
<td>10.79 %</td>
</tr>
<tr>
<td>Inline Method</td>
<td>8.43 %</td>
<td>Rename Method</td>
<td>5.97 %</td>
<td>Extract Method</td>
<td>8.13 %</td>
<td>Extract Class</td>
<td>10.31 %</td>
</tr>
<tr>
<td>Inline Temp Method</td>
<td>6.83 %</td>
<td>Pull Up Method</td>
<td>4.74 %</td>
<td>Rename Method</td>
<td>4.79 %</td>
<td>Rename Method</td>
<td>7.48 %</td>
</tr>
</tbody>
</table>
2. **Industrial case studies**

The rankings for the prioritization of techniques by company A are summarized as follows: First: Extract Method (16.21); Second: Pull Up Method (15.94); Third: Inline Class (15.8); Fourth: Extract Class (13.7); Fifth: Rename Method (11.01); Sixth: Inline Method (10.9); Seventh: Move Method (8.62); Eighth: Inline Temp Method (8.51). Table 6.72 summarizes the results.

Figure 6.44 shows the importance of each criterion as follows: code size (37.93), cohesion (30.72), coupling (16.21), and complexity (15.14).

<table>
<thead>
<tr>
<th>Refactoring Techniques</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract Method</td>
<td>16.21%</td>
</tr>
<tr>
<td>Pull Up Method</td>
<td>15.94%</td>
</tr>
<tr>
<td>Inline Class</td>
<td>15.8%</td>
</tr>
<tr>
<td>Extract Class</td>
<td>13.7%</td>
</tr>
<tr>
<td>Rename Method</td>
<td>11.01%</td>
</tr>
<tr>
<td>Inline Method</td>
<td>10.9%</td>
</tr>
<tr>
<td>Move Method</td>
<td>8.62%</td>
</tr>
<tr>
<td>Inline Temp Method</td>
<td>8.51%</td>
</tr>
</tbody>
</table>

The rankings for the prioritization of techniques by company B are summarized as follows: First: Extract Class (24); Second: Extract Method (19.99); Third: Move Method (12.44); Fourth: Inline Class (10.25); Fifth: Pull Up Method (9.79); Sixth: Inline Method
(8.83); Seventh: Rename Method (8.31); Eighth: Inline Temp Method (6.39). Table 6.73 summarizes the results.

Figure 6.45 shows the importance of each criterion as follows: cohesion (41.75), coupling (30.34), complexity (14.83), and code size (13.08).

The rankings for the prioritization of techniques by company C are summarized as follows: First: Extract Class (18.46); Second: Inline Class (17.98); Third: Extract Method (14.19); Fourth: Inline Method (13.89); Fifth: Inline Temp Method (10.58); Sixth: Move Method (10.4); Seventh: Pull Up Method (9.21); Eighth: Rename Method (5.29). Table 6.74 summarizes the results.

Figure 6.46 shows the importance of each criterion as follows: cohesion (38.37), coupling (38.76), code size (12.38), and complexity (10.49).
6.4.4.6 Observations (industrial case studies)

1. Considering all the criteria together, the Extract Class technique was ranked highest by companies B and C while company A ranked it in the fourth position. Company A ranked the Extract Method in the highest position.

2. The second position for each company was different. The second-highest position in A was Pull Up Method, in company B, it was Extract Method, and in company C it was Inline Class.

3. Inline Temp Method was ranked in the last position by companies A and B, while company C ranked the Rename Method in the last position.
4. Cohesion and coupling quality attributes were of the highest concern for companies B and C, while company A was most concerned with the code size and cohesion.

5. If we look at the refactoring techniques and consider each criterion individually as it is shown in tables 6.75, 6.76, and 6.77, we can see companies A and B ranked the Extract Class in the top position in the complexity attributes, while company C ranked the Extract Method at the top.

6. For the cohesion quality attribute, companies B and C ranked the Extract Class in the highest position, while company A ranked the Extract Method at the top.

7. For the coupling quality attribute, companies A, B, and C all ranked the Extract Class in the highest position.

8. For the code size, all the companies ranked the refactoring techniques differently. Companies A, B and C ranked the Pull Up Method, Inline Method, and Inline Class in the highest position, respectively.
## Table 6.75 Refactoring techniques based on each internal criterion by company A

<table>
<thead>
<tr>
<th>Refactoring Techniques</th>
<th>Complexity</th>
<th>Refactoring Techniques</th>
<th>Coupling</th>
<th>Refactoring Techniques</th>
<th>Chohesion</th>
<th>Refactoring Techniques</th>
<th>Code Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract Class</td>
<td>22.74 %</td>
<td>Extract Class</td>
<td>17.16 %</td>
<td>Extract Method</td>
<td>22.22 %</td>
<td>Pull Up Method</td>
<td>15.71 %</td>
</tr>
<tr>
<td>Extract Method</td>
<td>15.21 %</td>
<td>Pull Up Method</td>
<td>16.43 %</td>
<td>Inline Class</td>
<td>21.05 %</td>
<td>Inline Class</td>
<td>15.05 %</td>
</tr>
<tr>
<td>Pull Up Method</td>
<td>13.19 %</td>
<td>Extract Method</td>
<td>13.47 %</td>
<td>Rename Method</td>
<td>14.88 %</td>
<td>Extract Method</td>
<td>14.4 %</td>
</tr>
<tr>
<td>Inline Method</td>
<td>11.02 %</td>
<td>Rename Method</td>
<td>13.35 %</td>
<td>Extract Class</td>
<td>12.58 %</td>
<td>Inline Method</td>
<td>13.28 %</td>
</tr>
<tr>
<td>Move Method</td>
<td>10.49 %</td>
<td>Inline Class</td>
<td>11.98 %</td>
<td>Pull Up Method</td>
<td>8.46 %</td>
<td>Move Method</td>
<td>12.78 %</td>
</tr>
<tr>
<td>Inline Class</td>
<td>10.1 %</td>
<td>Inline Method</td>
<td>9.62 %</td>
<td>Inline Method</td>
<td>8.05 %</td>
<td>Inline Temp Method</td>
<td>12.25 %</td>
</tr>
<tr>
<td>Inline Temp Method</td>
<td>8.77 %</td>
<td>Inline Temp Method</td>
<td>8.38 %</td>
<td>Inline Temp Method</td>
<td>6.83 %</td>
<td>Extract Class</td>
<td>9.41 %</td>
</tr>
<tr>
<td>Rename Method</td>
<td>8.47 %</td>
<td>Move Method</td>
<td>8.59 %</td>
<td>Move Method</td>
<td>5.93 %</td>
<td>Rename Method</td>
<td>7.12 %</td>
</tr>
</tbody>
</table>

## Table 6.76 Refactoring techniques based on each internal criterion by company B

<table>
<thead>
<tr>
<th>Refactoring Techniques</th>
<th>Complexity</th>
<th>Refactoring Techniques</th>
<th>Coupling</th>
<th>Refactoring Techniques</th>
<th>Chohesion</th>
<th>Refactoring Techniques</th>
<th>Code Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract Class</td>
<td>28.73 %</td>
<td>Extract Class</td>
<td>28.85 %</td>
<td>Extract Class</td>
<td>24.23 %</td>
<td>Inline Method</td>
<td>18.41 %</td>
</tr>
<tr>
<td>Extract Method</td>
<td>21.4 %</td>
<td>Extract Class</td>
<td>23.53 %</td>
<td>Extract Method</td>
<td>20.19 %</td>
<td>Inline Class</td>
<td>18.38 %</td>
</tr>
<tr>
<td>Move Method</td>
<td>12.39 %</td>
<td>Move Method</td>
<td>11.92 %</td>
<td>Move Method</td>
<td>13.83 %</td>
<td>Extract Method</td>
<td>13.24 %</td>
</tr>
<tr>
<td>Pull Up Method</td>
<td>10.28 %</td>
<td>Pull Up Method</td>
<td>9.59 %</td>
<td>Inline Method</td>
<td>10.32 %</td>
<td>Extract Class</td>
<td>12.81 %</td>
</tr>
<tr>
<td>Rename Method</td>
<td>9.39 %</td>
<td>Rename Method</td>
<td>8.4 %</td>
<td>Pull Up Method</td>
<td>10.26 %</td>
<td>Inline Temp Method</td>
<td>11.57 %</td>
</tr>
<tr>
<td>Inline Method</td>
<td>7.03 %</td>
<td>Inline Class</td>
<td>6.55 %</td>
<td>Rename Method</td>
<td>8.39 %</td>
<td>Move Method</td>
<td>9.87 %</td>
</tr>
<tr>
<td>Inline Class</td>
<td>6.42 %</td>
<td>Inline Method</td>
<td>5.79 %</td>
<td>Inline Method</td>
<td>7.28 %</td>
<td>Pull Up Method</td>
<td>8.55 %</td>
</tr>
<tr>
<td>Inline Temp Method</td>
<td>4.35 %</td>
<td>Inline Temp Method</td>
<td>5.37 %</td>
<td>Inline Temp Method</td>
<td>5.5 %</td>
<td>Rename Method</td>
<td>7.17 %</td>
</tr>
</tbody>
</table>

## Table 6.77 Ranking the refactoring techniques based on each internal criterion by company C

<table>
<thead>
<tr>
<th>Refactoring Techniques</th>
<th>Complexity</th>
<th>Refactoring Techniques</th>
<th>Coupling</th>
<th>Refactoring Techniques</th>
<th>Chohesion</th>
<th>Refactoring Techniques</th>
<th>Code Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract Method</td>
<td>19.96 %</td>
<td>Extract Class</td>
<td>20.46 %</td>
<td>Extract Class</td>
<td>18.8 %</td>
<td>Inline Class</td>
<td>26 %</td>
</tr>
<tr>
<td>Inline Class</td>
<td>16.98 %</td>
<td>Inline Class</td>
<td>15.93 %</td>
<td>Inline Class</td>
<td>18.28 %</td>
<td>Inline Temp Method</td>
<td>15.16 %</td>
</tr>
<tr>
<td>Extract Class</td>
<td>13.7 %</td>
<td>Extract Class</td>
<td>15.41 %</td>
<td>Inline Method</td>
<td>14.85 %</td>
<td>Extract Class</td>
<td>14.31 %</td>
</tr>
<tr>
<td>Inline Method</td>
<td>13.4 %</td>
<td>Inline Method</td>
<td>13.4 %</td>
<td>Extract Method</td>
<td>13 %</td>
<td>Inline Method</td>
<td>12.33 %</td>
</tr>
<tr>
<td>Move Method</td>
<td>11.12 %</td>
<td>Pull Up Method</td>
<td>11.39 %</td>
<td>Inline Temp Method</td>
<td>11.82 %</td>
<td>Pull Up Method</td>
<td>9.94 %</td>
</tr>
<tr>
<td>Inline Temp Method</td>
<td>10.65 %</td>
<td>Move Method</td>
<td>11.17 %</td>
<td>Move Method</td>
<td>9.74 %</td>
<td>Move Method</td>
<td>9.32 %</td>
</tr>
<tr>
<td>Pull Up Method</td>
<td>9.64 %</td>
<td>Inline Temp Method</td>
<td>8.08 %</td>
<td>Pull Up Method</td>
<td>6.91 %</td>
<td>Extract Method</td>
<td>8.08 %</td>
</tr>
<tr>
<td>Rename Method</td>
<td>4.58 %</td>
<td>Rename Method</td>
<td>4.16 %</td>
<td>Rename Method</td>
<td>6.6 %</td>
<td>Rename Method</td>
<td>5.02 %</td>
</tr>
</tbody>
</table>
6.4.5 Secondly: Refactoring patterns based on external quality attributes

Before applying the AHP to refactoring patterns, I will highlight some of the previous studies that have investigated the impact of refactoring patterns on the external code quality attributes as follow:

Leitch and Stroulia [262] studied the refactoring effects on software maintenance effort and costs using dependency analysis.

Alshayeb [263] aimed to validate/invalidate the refactoring effects on some of the external quality attributes (adaptability, maintainability, understandability, reusability, and testability) in order to decide whether the cost and time put into refactoring are worthwhile. He concluded his study with results that stated that the refactoring does not necessarily improve these quality attributes.

Similarly, Raed and Li [264] used the hierarchal quality model to investigate the effect of refactoring activities on some software metrics such as reusability, flexibility, extendibility, and effectiveness. They found that not all the refactoring methods improve the quality factors.

Kataoka et al. [265] proposed a quantitative evaluation method to measure the maintainability enhancement by refactoring. They focused only on coupling metrics in order to quantify the refactoring effects.

Elish and Alshayeb [245,266] classified the refactoring techniques based on their effects on the internal and external quality attributes. The refactoring methods and the internal quality attributes were mentioned in section 6.4.4, while the external quality metrics were adaptability, completeness, maintainability, understandability, reusability, and testability.
Weber and Reichert [267] proposed 11 refactoring techniques to support the business process management at the operational level and allow the designer to improve the quality of the process model. They focused specifically on the process-aware information system model (PAIS) that provides schemes for process execution.

Moser et al. [268] conducted a case study in industrial projects and agile environment to analyze the impact of refactoring on the reusability attribute. They analyzed how often part of the software (i.e. classes, methods, etc.) is used in a product.

Stroulia and Leitch [269] proposed a method to estimate the expected software maintenance cost by predicting the return on investment (ROI) for the refactoring activity. The authors claim that this would increase the adoption of refactoring practices and improve the quality attributes of the software, such as performance and maintainability. Finally, Moser et al. [270] developed a model to identify a refactoring effort during the maintenance phase.

### 6.4.5.1 Proposed criteria for ranking the refactoring patterns

Similarly to internal refactoring, to rank the refactoring patterns it is necessary to identify the external quality attributes that are more desirable to the development team or the organization. Each project can have a different set of criteria and refactoring methods in order to be ranked and evaluated. In this section, we have chosen four external quality attributes to be the core criteria for the refactoring ranking:

- Reusability: defined as the capability for a component and subsystems to be suitable for use in more than one application, or in building other components, with little or no adaptation [271, 272].
• Flexibility: defined as “the ability of a system to adapt to varying environments and situations, and to cope with changes in business policies and rules. A flexible system is one that is easy to reconfigure or adapt in response to different user and system requirements” [273].

• Maintainability: defined as the ability of the system to accept changes with a degree of ease. These changes could be modifying a component or other attribute to correct faults, improve performance, or adapt to a new environment [274].

• Understandability: defined as the degree to which the meaning of a software component is clear to a user [271].

6.4.5.2 AHP-refactoring structure for the external attributes

The top level is the main objective: ranking the refactoring techniques; the second level is the criteria: reusability, flexibility, maintainability, and understandability; the third level is the alternatives: Extract Method, Inline Method, InlineTemp Method, Extract Class, Inline Class, Move Method, Pull Up Method, Rename Method.

Figure 6.74 illustrates the AHP structure for the problem.
Similarly, sheets of paper with appropriate AHP tables were handed to the all participants in order to save time and facilitate the process of comparison. The first page was dedicated to collecting general information about the evaluator, his/her experience, and the type and the level of his/her programming skills. The participants compared the criteria using the Saaty scale from 1-9. The participants were asked as examples:

- Which is more important: reusability or flexibility and by how much?
- Which is more important: reusability or maintainability and by how much?
- Which is more important: reusability or flexibility and by how much?
- Which is more important: reusability or understandability and by how much?
- Which is more important: flexibility or maintainability and by how much?
- Which is more important: flexibility or understandability and by how much?
- Which is more important: maintainability or understandability and by how much?

After finishing the criteria comparisons, the participants had to evaluate all the refactoring techniques based on each criterion, exactly how it has been shown previously with the internal attributes.
6.4.5.3 AHP evaluation results for the refactoring pattern

1. Educational case studies

For Team 1, the rankings for the refactoring techniques based on all criteria (i.e. reusability, flexibility, maintainability and understandability) are summarized as follows: First: Extract Class (18.33); Second: Rename Method (15.17); Third: Pull Up Method (14.13); Fourth: Extract Method (13.95); Fifth: Inline Class (12.81); Sixth: Move Method (12.25); Seventh: Inline Method (6.74); Eighth: Inline Temp Method (6.62). Table 6.78 summarizes the results.

Figure 6.48 shows the importance of each criterion as follows: understandability (35.70), maintainability (30.09), reusability (22.42), and flexibility (11.80).

<table>
<thead>
<tr>
<th>Refactoring Techniques</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract Class</td>
<td>18.33%</td>
</tr>
<tr>
<td>Rename Method</td>
<td>15.17%</td>
</tr>
<tr>
<td>Pull Up Method</td>
<td>14.13%</td>
</tr>
<tr>
<td>Extract Method</td>
<td>13.95%</td>
</tr>
<tr>
<td>Inline Class</td>
<td>12.81%</td>
</tr>
<tr>
<td>Move Method</td>
<td>12.25%</td>
</tr>
<tr>
<td>Inline Method</td>
<td>6.74%</td>
</tr>
<tr>
<td>Inline Temp Method</td>
<td>6.62%</td>
</tr>
</tbody>
</table>

Table 6.78 Ranking the refactoring techniques by Team 1 (external attributes)

Figure 6.48 Importance of the external criteria for the refactoring by Team 1
The rankings for the prioritization of techniques by Team 2 is summarized as follows: First: Extract Class (16.66); Second: Extract Method (16.37); Third: Rename Method (15.41); Fourth: Pull Up Method (15.16); Fifth: Move Method (12.74); Sixth: Inline Temp Method (8.07); Seventh: Inline Method (7.92); Eighth: Inline Class (7.66). Table 6.79 summarizes the results.

Figure 6.49 shows the importance of each criterion as follows: maintainability (30.85), understandability (29.7), flexibility (21.77), and reusability (17.67).

<table>
<thead>
<tr>
<th>Refactoring Techniques</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract Class</td>
<td>16.66 %</td>
</tr>
<tr>
<td>Extract Method</td>
<td>16.37 %</td>
</tr>
<tr>
<td>Rename Method</td>
<td>15.41 %</td>
</tr>
<tr>
<td>Pull Up Method</td>
<td>15.16 %</td>
</tr>
<tr>
<td>Move Method</td>
<td>12.74 %</td>
</tr>
<tr>
<td>Inline Temp Method</td>
<td>8.07 %</td>
</tr>
<tr>
<td>Inline Method</td>
<td>7.92 %</td>
</tr>
<tr>
<td>Inline Class</td>
<td>7.66 %</td>
</tr>
</tbody>
</table>

Figure 6.49 Importance of the external criteria for the refactoring by Team 2
6.4.5.4 Observations (educational case studies)

1. Considering all the criteria together, the Extract Class technique was ranked in the highest position by both teams, Team 1 and Team 2.

2. The Rename Method was ranked in advanced positions. Team 1 ranked it in the second position and Team 2 in the second position.

3. The understandability and maintainability quality attributes were considered the most important by both teams.

4. If we look at the refactoring techniques considering each criterion individually, we can see both teams ranked the Rename Method in the top position in the understandability attributes. For other criteria, each team has ranked the refactoring techniques differently. See tables 6.80 and 6.81.

5. For the reusability quality attribute, Team 1 ranked the Extract Class in the highest position, while Team 2 ranked the Extract Method at the top.

6. For the flexibility quality attribute, Team 1 ranked the Rename in the highest position, while Team 2 ranked the Extract Class at the top.

7. For the maintainability quality attribute, Team 1 ranked the Extract Class in the highest position, while Team 2 ranked the Pull Up Method at the top.

8. We can note the Extract Class was in the highest position for Team 1 in reusability and maintainability as individual criterion, while Team 2 considered it in the top only with the flexibility criterion.
Table 6.80 Refactoring techniques based on each external criterion by Team 1

<table>
<thead>
<tr>
<th>Refactoring Techniques</th>
<th>Resuability</th>
<th>Refactoring Techniques</th>
<th>Maintainability</th>
<th>Refactoring Techniques</th>
<th>Flexibility</th>
<th>Refactoring Techniques</th>
<th>Understandability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract Class</td>
<td>22.43 %</td>
<td>Extract Class</td>
<td>23.8 %</td>
<td>Rename Method</td>
<td>17.32 %</td>
<td>Rename Method</td>
<td>25.13 %</td>
</tr>
<tr>
<td>Extract Method</td>
<td>15.53 %</td>
<td>Extract Method</td>
<td>16.82 %</td>
<td>Extract Class</td>
<td>15.45 %</td>
<td>Extract Class</td>
<td>15.47 %</td>
</tr>
<tr>
<td>Pull Up Method</td>
<td>14.54 %</td>
<td>Pull Up Method</td>
<td>13.34 %</td>
<td>Pull Up Method</td>
<td>14.57 %</td>
<td>Inline Class</td>
<td>13.26 %</td>
</tr>
<tr>
<td>Move Method</td>
<td>13.89 %</td>
<td>Inline Class</td>
<td>10.87 %</td>
<td>Inline Class</td>
<td>11.99 %</td>
<td>Pull Up Method</td>
<td>11.91 %</td>
</tr>
<tr>
<td>Inline Class</td>
<td>11.46 %</td>
<td>Move Method</td>
<td>10.75 %</td>
<td>Extract Method</td>
<td>11.94 %</td>
<td>Move Method</td>
<td>11.75 %</td>
</tr>
<tr>
<td>Rename Method</td>
<td>9.72 %</td>
<td>Rename Method</td>
<td>8.32 %</td>
<td>Inline Temp Method</td>
<td>10.34 %</td>
<td>Extract Method</td>
<td>9.44 %</td>
</tr>
<tr>
<td>Inline Temp Method</td>
<td>6.35 %</td>
<td>Inline Temp Method</td>
<td>8.25 %</td>
<td>Inline Method</td>
<td>9.43 %</td>
<td>Inline Method</td>
<td>7.04 %</td>
</tr>
<tr>
<td>Inline Method</td>
<td>6.09 %</td>
<td>Inline Method</td>
<td>7.86 %</td>
<td>Move Method</td>
<td>8.95 %</td>
<td>Inline Temp Method</td>
<td>6.00 %</td>
</tr>
</tbody>
</table>

Table 6.81 Refactoring techniques based on each external criterion by Team 2

<table>
<thead>
<tr>
<th>Refactoring Techniques</th>
<th>Resuability</th>
<th>Refactoring Techniques</th>
<th>Maintainability</th>
<th>Refactoring Techniques</th>
<th>Flexibility</th>
<th>Refactoring Techniques</th>
<th>Understandability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract Method</td>
<td>22.43 %</td>
<td>Pull Up Method</td>
<td>21.75 %</td>
<td>Extract Class</td>
<td>20.74 %</td>
<td>Rename Method</td>
<td>22.82 %</td>
</tr>
<tr>
<td>Extract Class</td>
<td>15.53 %</td>
<td>Rename Method</td>
<td>17.18 %</td>
<td>Pull Up Method</td>
<td>14.92 %</td>
<td>Extract Method</td>
<td>17.39 %</td>
</tr>
<tr>
<td>Pull Up Method</td>
<td>14.54 %</td>
<td>Extract Method</td>
<td>14.4 %</td>
<td>Move Method</td>
<td>14.08 %</td>
<td>Move Method</td>
<td>16.72 %</td>
</tr>
<tr>
<td>Move Method</td>
<td>13.89 %</td>
<td>Extract Class</td>
<td>14.03 %</td>
<td>Extract Method</td>
<td>12.92 %</td>
<td>Extract Class</td>
<td>13.41 %</td>
</tr>
<tr>
<td>Rename Method</td>
<td>11.46 %</td>
<td>Inline Temp Method</td>
<td>8.61 %</td>
<td>Inline Method</td>
<td>10.52 %</td>
<td>Pull Up Method</td>
<td>8.4 %</td>
</tr>
<tr>
<td>Inline Temp Method</td>
<td>9.72 %</td>
<td>Move Method</td>
<td>8.6 %</td>
<td>Inline Class</td>
<td>10.38 %</td>
<td>Inline Temp Method</td>
<td>7.67 %</td>
</tr>
<tr>
<td>Inline Method</td>
<td>6.35 %</td>
<td>Inline Class</td>
<td>8 %</td>
<td>Inline Temp Method</td>
<td>8.27 %</td>
<td>Inline Method</td>
<td>7.27 %</td>
</tr>
<tr>
<td>Inline Class</td>
<td>6.09 %</td>
<td>Inline Method</td>
<td>7.43 %</td>
<td>Rename Method</td>
<td>8.18 %</td>
<td>Inline Class</td>
<td>8.31 %</td>
</tr>
</tbody>
</table>
2. Industrial case studies

The rankings for the prioritization of techniques by company A are summarized as follows: First: Inline Class (15.71); Second: Extract Method (15.05); Third: Extract Class (14.4); Fourth: Move Method (13.28); Fifth: Pull Up Method (12.78); Sixth: Inline Method (12.25); Seventh: Inline Temp Method (9.41); Eighth: Rename Method (7.12). Table 6.82 summarizes the results.

Figure 6.50 shows the importance of each criterion as follows: flexibility (35.97), maintainability (28.51), reusability (24.71), and understandability (10.81).

<table>
<thead>
<tr>
<th>Refactoring Techniques</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inline Class</td>
<td>15.71 %</td>
</tr>
<tr>
<td>Extract Method</td>
<td>15.05 %</td>
</tr>
<tr>
<td>Extract Class</td>
<td>14.4 %</td>
</tr>
<tr>
<td>Move Method</td>
<td>13.28 %</td>
</tr>
<tr>
<td>Pull Up Method</td>
<td>12.78 %</td>
</tr>
<tr>
<td>Inline Method</td>
<td>12.25 %</td>
</tr>
<tr>
<td>Inline Temp Method</td>
<td>9.41 %</td>
</tr>
<tr>
<td>Rename Method</td>
<td>7.12 %</td>
</tr>
</tbody>
</table>

Table 6.82 Ranking the refactoring techniques by company A (external attributes)

The rankings for the prioritization of techniques by company B are summarized as follows: First: Extract Class (27.84); Second: Extract Method (18.26); Third: Inline Class (11.07); Fourth: Move Method (10.37); Fifth: Inline Method (8.9); Sixth: Rename Method (8.81); Seventh: Pull Up Method (8.76); Eighth: Inline Temp Method (5.99). Table 6.83 summarizes the results.
Figure 6.51 shows the importance of each criterion as follows: understandability (30.5), reusability (29.42), maintainability (26.97), and flexibility (13.11).

<table>
<thead>
<tr>
<th>Refactoring Techniques</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract Class</td>
<td>27.84 %</td>
</tr>
<tr>
<td>Extract Method</td>
<td>18.26 %</td>
</tr>
<tr>
<td>Inline Class</td>
<td>11.07 %</td>
</tr>
<tr>
<td>Move Method</td>
<td>10.37 %</td>
</tr>
<tr>
<td>Inline Method</td>
<td>8.9 %</td>
</tr>
<tr>
<td>Rename Method</td>
<td>8.81 %</td>
</tr>
<tr>
<td>Pull Up Method</td>
<td>8.76 %</td>
</tr>
<tr>
<td>Inline Temp Method</td>
<td>5.99 %</td>
</tr>
</tbody>
</table>

The rankings for the prioritization of techniques by company Care summarized as follows: First: Extract Class (21.7); Second: Inline Class (17.38); Third: Inline Method (11.83); Fourth: Extract Method (11.47); Fifth: Pull Up Method (10.86); Sixth: Move Method (9.7); Seventh: Inline Temp Method (9.35); Eighth: Rename Method (7.71). Table 6.84 summarizes the results.

Figure 6.52 shows the importance of each criterion as follows: reusability (42.19), maintainability (26.34), understandability (18.76), and flexibility (12.71).
6.4.5.5 Observations (industrial case studies)

1. Considering all the criteria together, the Extract Class technique was ranked in the highest position by companies B and C, while company A ranked the Inline Class in the highest position.

2. Extract Method was ranked in second position by companies A and B, while company C ranked the Inline Class in the second position.

3. The Rename Method was ranked in last positions by companies A and C, while company B ranked the Inline Temp Method in the last position. Also, A and C ranked the Inline Temp Method in the penultimate position.

4. Each company has ranked the quality attributes differently. Company A ranked maintainability and flexibility in the two top positions. Company B considered reusability and understandability to be the top ones. Company C considered reusability and maintainability the highest concerns. As result, companies A and
C shared the same concerns about maintainability and ranked it high. On the other hand, companies B and C shared the concerns about reusability and ranked it to be one of the top criteria.

5. If we look at the refactoring techniques considering each criterion individually, we can see company B ranked the Extract Class in the top position in all the quality attributes: reusability, flexibility, maintainability, and understandability. Company C also ranked the Extract Class in the top position in terms of reusability and understandability, while company A ranked it in the top only in terms of maintainability. See tables 6.85, 6.86, and 6.87.

6. For the reusability quality attribute, Team 1 ranked the Extract Class in the highest position, while Team 2 ranked the Extract Method at the top.

7. Company C ranked the Inline Class in the first position in terms of flexibility and maintainability, while company A ranked it at the top in reusability.

8. Company A ranked the Extract Method at the top in flexibility and the Move Method at the top in understandability.
Table 6.85 Refactoring techniques based on each external criterion by company A

<table>
<thead>
<tr>
<th>Refactoring Techniques</th>
<th>Resuability</th>
<th>Refactoring Techniques</th>
<th>Maintainability</th>
<th>Refactoring Techniques</th>
<th>Flexibility</th>
<th>Refactoring Techniques</th>
<th>Understandability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inline Class</td>
<td>17.21 %</td>
<td>Extract Class</td>
<td>24.05 %</td>
<td>Extract Method</td>
<td>21.27 %</td>
<td>Move Method</td>
<td>15.71 %</td>
</tr>
<tr>
<td>Inline Method</td>
<td>16.62 %</td>
<td>Inline Class</td>
<td>19.87 %</td>
<td>Move Method</td>
<td>16.16 %</td>
<td>Pull Up Method</td>
<td>15.05 %</td>
</tr>
<tr>
<td>Pull Up Method</td>
<td>16.61 %</td>
<td>Inline Method</td>
<td>15.96 %</td>
<td>Pull Up Method</td>
<td>13.4 %</td>
<td>Extract Class</td>
<td>14.4 %</td>
</tr>
<tr>
<td>Extract Method</td>
<td>14.71 %</td>
<td>Inline Temp Method</td>
<td>13.97 %</td>
<td>Inline Class</td>
<td>13.08 %</td>
<td>Rename Method</td>
<td>13.28 %</td>
</tr>
<tr>
<td>Move Method</td>
<td>12.59 %</td>
<td>Extract Method</td>
<td>10.54 %</td>
<td>Extract Class</td>
<td>12.02 %</td>
<td>Inline Class</td>
<td>12.78 %</td>
</tr>
<tr>
<td>Extract Class</td>
<td>8.91 %</td>
<td>Move Method</td>
<td>7.47 %</td>
<td>Rename Method</td>
<td>6.67 %</td>
<td>Inline Temp Method</td>
<td>12.25 %</td>
</tr>
<tr>
<td>Rename Method</td>
<td>7.09 %</td>
<td>Pull Up Method</td>
<td>5.19 %</td>
<td>Inline Temp Method</td>
<td>8.49 %</td>
<td>Inline Method</td>
<td>9.41 %</td>
</tr>
<tr>
<td>Inline Temp Method</td>
<td>6.47 %</td>
<td>Rename Method</td>
<td>2.95 %</td>
<td>Inline Method</td>
<td>6.91 %</td>
<td>Extract Method</td>
<td>7.12 %</td>
</tr>
</tbody>
</table>

Table 6.86 Ranking the refactoring techniques based on each external criterion by company B

<table>
<thead>
<tr>
<th>Refactoring Techniques</th>
<th>Resuability</th>
<th>Refactoring Techniques</th>
<th>Maintainability</th>
<th>Refactoring Techniques</th>
<th>Flexibility</th>
<th>Refactoring Techniques</th>
<th>Understandability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract Class</td>
<td>27.83 %</td>
<td>Extract Class</td>
<td>31.17 %</td>
<td>Extract Class</td>
<td>19.86 %</td>
<td>Extract Class</td>
<td>30.21 %</td>
</tr>
<tr>
<td>Extract Method</td>
<td>17.08 %</td>
<td>Extract Method</td>
<td>20.93 %</td>
<td>Extract Method</td>
<td>15.87 %</td>
<td>Extract Method</td>
<td>18.77 %</td>
</tr>
<tr>
<td>Inline Class</td>
<td>13.8 %</td>
<td>Inline Class</td>
<td>10.81 %</td>
<td>Move Method</td>
<td>14.44 %</td>
<td>Rename Method</td>
<td>11.95 %</td>
</tr>
<tr>
<td>Move Method</td>
<td>9.74 %</td>
<td>Move Method</td>
<td>8.97 %</td>
<td>Inline Class</td>
<td>13.03 %</td>
<td>Pull Up Method</td>
<td>10.02 %</td>
</tr>
<tr>
<td>Inline Method</td>
<td>9.62 %</td>
<td>Rename Method</td>
<td>7.88 %</td>
<td>Inline Method</td>
<td>12.03 %</td>
<td>Move Method</td>
<td>9.58 %</td>
</tr>
<tr>
<td>Pull Up Method</td>
<td>8.62 %</td>
<td>Pull Up Method</td>
<td>7.84 %</td>
<td>Inline Temp Method</td>
<td>9.4 %</td>
<td>Inline Method</td>
<td>7.58 %</td>
</tr>
<tr>
<td>Rename Method</td>
<td>7.71 %</td>
<td>Inline Method</td>
<td>7.16 %</td>
<td>Pull Up Method</td>
<td>8.29 %</td>
<td>Inline Class</td>
<td>7.1 %</td>
</tr>
<tr>
<td>Inline Temp Method</td>
<td>5.61 %</td>
<td>Inline Temp Method</td>
<td>5.25 %</td>
<td>Rename Method</td>
<td>6.98 %</td>
<td>Inline Temp Method</td>
<td>4.8 %</td>
</tr>
</tbody>
</table>

Table 6.87 Ranking the refactoring techniques based on each external criterion by company C

<table>
<thead>
<tr>
<th>Refactoring Techniques</th>
<th>Resuability</th>
<th>Refactoring Techniques</th>
<th>Maintainability</th>
<th>Refactoring Techniques</th>
<th>Flexibility</th>
<th>Refactoring Techniques</th>
<th>Understandability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract Class</td>
<td>25.04 %</td>
<td>Inline Class</td>
<td>23.8 %</td>
<td>Inline Class</td>
<td>19.86 %</td>
<td>Extract Class</td>
<td>30.21 %</td>
</tr>
<tr>
<td>Extract Method</td>
<td>14.54 %</td>
<td>Extract Class</td>
<td>16.82 %</td>
<td>Extract Class</td>
<td>15.87 %</td>
<td>Inline Class</td>
<td>18.77 %</td>
</tr>
<tr>
<td>Move Method</td>
<td>12.53 %</td>
<td>Inline Method</td>
<td>13.34 %</td>
<td>Move Method</td>
<td>14.44 %</td>
<td>Inline Method</td>
<td>11.95 %</td>
</tr>
<tr>
<td>Pull Up Method</td>
<td>11.78 %</td>
<td>Pull Up Method</td>
<td>10.87 %</td>
<td>Inline Temp Method</td>
<td>13.03 %</td>
<td>Extract Method</td>
<td>10.02 %</td>
</tr>
<tr>
<td>Inline Class</td>
<td>10.34 %</td>
<td>Inline Temp Method</td>
<td>10.75 %</td>
<td>Inline Method</td>
<td>12.03 %</td>
<td>Pull Up Method</td>
<td>9.58 %</td>
</tr>
<tr>
<td>Inline Method</td>
<td>10.22 %</td>
<td>Rename Method</td>
<td>8.32 %</td>
<td>Rename Method</td>
<td>9.4 %</td>
<td>Inline Temp Method</td>
<td>7.58 %</td>
</tr>
<tr>
<td>Inline Temp Method</td>
<td>7.89 %</td>
<td>Extract Method</td>
<td>8.25 %</td>
<td>Extract Method</td>
<td>8.29 %</td>
<td>Rename Method</td>
<td>7.1 %</td>
</tr>
<tr>
<td>Rename Method</td>
<td>7.66 %</td>
<td>Move Method</td>
<td>7.86 %</td>
<td>Pull Up Method</td>
<td>6.98 %</td>
<td>Move Method</td>
<td>4.8 %</td>
</tr>
</tbody>
</table>
6.4.6 Refactoring experience by students

The participants in the educational studies were required to practice these refactoring patterns through developing their assigned project. Also, they were required to note the refactoring’s impact on the code quality attributes. Here are some examples the refactoring used.

Table 6.88 shows some examples of the refactoring done in the projects and the benefits obtained.

<table>
<thead>
<tr>
<th>Refactoring Techniques</th>
<th>Where/ implementing</th>
<th>Advantages</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pull up Method</strong></td>
<td>Visibility of configuration page</td>
<td>Less code duplication/better code readability.</td>
</tr>
<tr>
<td></td>
<td>Client Issuepanel.cs</td>
<td>Number of pages that are required for operation decreased</td>
</tr>
<tr>
<td></td>
<td>Manipulating the issue main page to use that for delete/edit/update</td>
<td></td>
</tr>
<tr>
<td><strong>Extract Method</strong></td>
<td>Handled result of login child window</td>
<td>Improved readability of code</td>
</tr>
<tr>
<td><strong>Inline class</strong></td>
<td>Removed common class for login/logout operation</td>
<td>Reduced code size coupling was reduced between classes containing login and logout methods</td>
</tr>
<tr>
<td><strong>Move Method</strong></td>
<td>Login validation</td>
<td>Increased maintainability and understandability</td>
</tr>
<tr>
<td><strong>Rename Method</strong></td>
<td>Changed name of login completion to login event handler method</td>
<td>Provided more understandability and maintainability</td>
</tr>
<tr>
<td></td>
<td>Rename function and button</td>
<td>Improved code readability</td>
</tr>
<tr>
<td><strong>Abstraction</strong></td>
<td>Server GetJoinedIssue Quarry</td>
<td>Faster access and easy filtering</td>
</tr>
<tr>
<td></td>
<td>Stored procedure</td>
<td>Easier of maintainability</td>
</tr>
<tr>
<td><strong>Push down</strong></td>
<td>Client IssuePanel.cs</td>
<td>Uniform methods and ease of access</td>
</tr>
<tr>
<td><strong>Inline Method</strong></td>
<td>Implementing the method to be one line while loading the filter.</td>
<td>Reduced code size</td>
</tr>
<tr>
<td></td>
<td>Combining the methods of delete and edit to the method of authority</td>
<td>Improved the code reusability, More maintainable, Reduced code size</td>
</tr>
<tr>
<td><strong>Extract class</strong></td>
<td>Creating site-master page</td>
<td>Improved the code reusability</td>
</tr>
<tr>
<td></td>
<td>Conducting the user authentication in the site-master page</td>
<td>Increased the cohesion and decreased the coupling</td>
</tr>
<tr>
<td></td>
<td>Creating other web forms from the site-master page</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Designing address class</td>
<td></td>
</tr>
</tbody>
</table>
In the following tables 6.89 and 6.90, Team 1 and Team 2 have reported the number of refactoring techniques that have been used in each iteration.

<table>
<thead>
<tr>
<th>Table 6.89 Number of refactoring was used for each iteration by Team 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Iteration</strong></td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>Extract Method</td>
</tr>
<tr>
<td>Inline Method</td>
</tr>
<tr>
<td>InlineTemp Method</td>
</tr>
<tr>
<td>Extract Class</td>
</tr>
<tr>
<td>Inline Class</td>
</tr>
<tr>
<td>Move Method</td>
</tr>
<tr>
<td>Pull Up Method</td>
</tr>
<tr>
<td>Rename Method</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 6.90 Number of refactoring was used for each iteration by Team 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Iteration</strong></td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>Extract Method</td>
</tr>
<tr>
<td>Inline Method</td>
</tr>
<tr>
<td>InlineTemp Method</td>
</tr>
<tr>
<td>Extract Class</td>
</tr>
<tr>
<td>Inline Class</td>
</tr>
<tr>
<td>Move Method</td>
</tr>
<tr>
<td>Pull Up Method</td>
</tr>
<tr>
<td>Rename Method</td>
</tr>
</tbody>
</table>

In the following sections, we will observe the impact of the refactoring techniques on the code internal and external quality attributes. All participants were required to use (+) to denote an increase, (-) to denote a decrease, (0) to show no changes on the measures or not used.
6.4.7. Observations of the internal impact of refactoring

Tables 6.91 and 6.92 show the impact of the refactoring on the internal quality attributes reported by Team 1 and Team 2.

Team 1:

Table 6.91 Refactoring impact on the internal attributes by Team 1

<table>
<thead>
<tr>
<th></th>
<th>Complexity</th>
<th>Cohesion</th>
<th>Coupling</th>
<th>Code Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract Method</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Inline Method</td>
<td>0</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>InlineTemp Method</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Extract Class</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Inline Class</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Move Method</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Pull Up Method</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Rename Method</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Team 2:

Table 6.92 Refactoring impact on the internal attributes by Team 2

<table>
<thead>
<tr>
<th></th>
<th>Complexity</th>
<th>Cohesion</th>
<th>Coupling</th>
<th>Code Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract Method</td>
<td>-</td>
<td>+</td>
<td>0</td>
<td>+</td>
</tr>
<tr>
<td>Inline Method</td>
<td>+</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>InlineTemp Method</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>Extract Class</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Inline Class</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Move Method</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>0</td>
</tr>
<tr>
<td>Pull Up Method</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Rename Method</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
6.4.8. Observations on the external impact of refactoring

Tables 6.93 and 6.94 show the impact of the refactoring on the external quality attributes reported by Team 1 and Team 2.

Team 1:

Table 6.93 Refactoring impact on the external attributes by Team 1

<table>
<thead>
<tr>
<th></th>
<th>Reusability</th>
<th>Flexibility</th>
<th>Maintainability</th>
<th>Understandability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract Method</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Inline Method</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>InlineTemp Method</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Extract Class</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Inline Class</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>0</td>
</tr>
<tr>
<td>Move Method</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>0</td>
</tr>
<tr>
<td>Pull Up Method</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Rename Method</td>
<td>0</td>
<td>0</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

Team 2:

Table 6.94 Refactoring impact on the external attributes by Team 2

<table>
<thead>
<tr>
<th></th>
<th>Reusability</th>
<th>Flexibility</th>
<th>Maintainability</th>
<th>Understandability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract Method</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Inline Method</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>InlineTemp Method</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>Extract Class</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Inline Class</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Move Method</td>
<td>0</td>
<td>0</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Pull Up Method</td>
<td>+</td>
<td>0</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Rename Method</td>
<td>0</td>
<td>0</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>
6.4.9 Refactoring validation

From the previous AHP evaluation conducted by students and experts, we found out that there are three refactoring techniques that have received high rankings: Extract Class, Inline Class, and Extract Method.

In this section, we collected technical information from two educational studies and two industrial studies to validate the AHP evaluation results obtained previously. The collected information included the following:

- The impact of the proposed refactoring techniques on the internal and external quality attributes.
- How many times each refactoring technique was used for each iteration in each case study.

In addition, the participants were required to count the time spent for the refactoring before and after using AHP.

6.4.9.1 Number of times using the refactoring techniques

Tables 6.95 and 6.96 summarize how many times each of the refactoring techniques were used in each iteration by companies A and B.

**Company A**

<p>| Table 6.95 Number of times refactoring was used for each iteration by company A |
|-------------------|----------------|----------------|----------------|----------------|----------------|----------------|</p>
<table>
<thead>
<tr>
<th></th>
<th>Iteration 1</th>
<th>Iteration 2</th>
<th>Iteration 3</th>
<th>Iteration 4</th>
<th>Iteration 5</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract Method</td>
<td>27</td>
<td>34</td>
<td>21</td>
<td>49</td>
<td>41</td>
<td>172</td>
</tr>
<tr>
<td>Inline Method</td>
<td>5</td>
<td>13</td>
<td>17</td>
<td>8</td>
<td>22</td>
<td>65</td>
</tr>
<tr>
<td>InlineTemp Method</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>7</td>
<td>5</td>
<td>16</td>
</tr>
<tr>
<td>Extract Class</td>
<td>18</td>
<td>11</td>
<td>12</td>
<td>15</td>
<td>9</td>
<td>65</td>
</tr>
<tr>
<td>Inline Class</td>
<td>18</td>
<td>10</td>
<td>7</td>
<td>11</td>
<td>3</td>
<td>49</td>
</tr>
<tr>
<td>Move Method</td>
<td>8</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>20</td>
<td>34</td>
</tr>
<tr>
<td>Pull Up Method</td>
<td>21</td>
<td>17</td>
<td>9</td>
<td>27</td>
<td>13</td>
<td>87</td>
</tr>
<tr>
<td>Rename Method</td>
<td>11</td>
<td>0</td>
<td>17</td>
<td>22</td>
<td>5</td>
<td>55</td>
</tr>
</tbody>
</table>
Company B

Table 6.96 Number of times refactoring was used for each iteration by company B

<table>
<thead>
<tr>
<th>Method</th>
<th>Iteration 1</th>
<th>Iteration 2</th>
<th>Iteration 3</th>
<th>Iteration 4</th>
<th>Iteration 5</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract Method</td>
<td>9</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>18</td>
</tr>
<tr>
<td>Inline Method</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>InlineTemp Method</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Extract Class</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>18</td>
</tr>
<tr>
<td>Inline Class</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Move Method</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Pull Up Method</td>
<td>7</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>17</td>
</tr>
<tr>
<td>Rename Method</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
</tbody>
</table>

6.4.9.2 Refactoring impact on the internal quality attributes

Tables 6.97 and 6.98 summarize the internal impacts for each refactoring technique for companies A and B.

Company A:

Table 6.97 Refactoring impact on the internal attributes by company A

<table>
<thead>
<tr>
<th>Method</th>
<th>Complexity</th>
<th>Cohesion</th>
<th>Coupling</th>
<th>Code Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract Method</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Inline Method</td>
<td>-</td>
<td>0</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>InlineTemp Method</td>
<td>+</td>
<td>-</td>
<td>0</td>
<td>+</td>
</tr>
<tr>
<td>Extract Class</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Inline Class</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Move Method</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Pull Up Method</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Rename Method</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Company B

Table 6.98 Refactoring impact on the internal attributes by company B

<table>
<thead>
<tr>
<th>Method</th>
<th>Complexity</th>
<th>Cohesion</th>
<th>Coupling</th>
<th>Code Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract Method</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Inline Method</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>InlineTemp Method</td>
<td>+</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>Extract Class</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Inline Class</td>
<td>+</td>
<td>0</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Move Method</td>
<td>+</td>
<td>-</td>
<td>0</td>
<td>+</td>
</tr>
<tr>
<td>Pull Up Method</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Rename Method</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
6.4.9.3 Refactoring impact on the external quality attributes

Table 6.99 and 6.100 summarize the external impacts for each refactoring technique for companies A and B.

Company A:

Table 6.99 Refactoring impact on the external attributes by company A

<table>
<thead>
<tr>
<th></th>
<th>Reusability</th>
<th>Flexibility</th>
<th>Maintainability</th>
<th>Understandability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract Method</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Inline Method</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>InlineTemp Method</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Extract Class</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Inline Class</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Move Method</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Pull Up Method</td>
<td>0</td>
<td>0</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Rename Method</td>
<td>NA</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

Company B:

Table 6.100 Refactoring impact on the external attributes by company B

<table>
<thead>
<tr>
<th></th>
<th>Reusability</th>
<th>Flexibility</th>
<th>Maintainability</th>
<th>Understandability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract Method</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Inline Method</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>InlineTemp Method</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Extract Class</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Inline Class</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>Move Method</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>Pull Up Method</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Rename Method</td>
<td>0</td>
<td>0</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>
6.4.9.4 AHP-refactoring impact on time

Both companies were asked to provide the time spent before and after using the AHP.

Table 6.101 shows that the time was reduced.

<table>
<thead>
<tr>
<th></th>
<th>Time before AHP</th>
<th>After AHP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company A</td>
<td>3 Days</td>
<td>2 Days</td>
</tr>
<tr>
<td>Company B</td>
<td>12 hours</td>
<td>7 hours</td>
</tr>
</tbody>
</table>

6.4.9.5 Observations from the validation

- From the AHP evaluation results, Extract Class and Extract Method were mostly ranked in the top positions in the internal and external quality attributes. Tables 6.95 and 6.99 show that Extract Method and Extract Class are commonly used and have positive effects (as seen in tables 6.97, 6.98, 6.99 and 6.100). In Company A, the most refactoring techniques were used: Extract Method (172), Pull Up Method (87), and Extract Class and Inline Method (65). In Company B, the fewest refactoring techniques were used: Extract Method and Extract Class (18), and Pull Up Method (17).

- Extract Class showed a positive impact on all the internal quality attributes in company A. The complexity, coupling and code size were reduced while the cohesion increased. In company B, the complexity and coupling reduced as well, but the code size increased. The cohesion increased as positive result. For both
companies, Extract Method showed a positive impact on all the internal quality attributes, except the code size increased.

- For the external quality, Extract Method had a positive impact on all the attributes in both companies’ results. The reusability, flexibility, and understandability increased; the maintainability effort was reduced. For both companies, Extract Class increased reusability, flexibility, and understandability, which is a good result. However, the maintainability also increased, which is a negative impact.

- Pull Up Method was used many times by the two companies, even though it was not showing significant results by the AHP evaluation. Also, it showed a positive impact on both internal and external quality attributes generally.

- After narrowing the use of refactoring patterns and ranking them using AHP, the time for refactoring for company A was reduced from 3 days to 2 days, while the refactoring time for company B was reduced from 12 hours to 7 hours.
6.5 Test-Driven Development

This chapter describes two proposed uses of AHP in testing. An introduction of the concept of TDD and the current research in the field is presented as well. This chapter shows AHP-testing hierarchy structures that include some important criteria for testing techniques and release indicators. The findings and results of educational and industrial case studies are also presented and discussed.

6.5.1 Introduction

Test-Driven Development (TDD) is the most important XP practice, where unit test cases are incrementally written before implementation takes place. It was first described in detail by Beck in [275] and successfully adapted in numerous agile projects. TDD is a highly recommended concept for software development because of a range of benefits such as increasing programming productivity and speed, producing high design quality, and reducing the work required for fixing defects [276].

The majority of tests in XP projects are automated and must pass every time. The tests are written just before starting the code. The test automation in XP helps to save time and effort, and reduce the long term cost of testing. It is preferable to perform regressions that are continuously changing in a shorter time and give us the ability to run the test simultaneously on different machines. In addition, the testing tool can be used for both unit and acceptance testing [277].

Automated testing is undoubtedly the heart of TDD, but it can be more expensive than manual testing [278]. So XP team should not do automated tests for the sake of automation alone. There are situations when you need to conduct semi-automated tests or even manual ones. JoEllen, an agile tester from VersionOne, believes that if a test is only
performed once it should not be automated “One-time tasks and exploratory testing/edge cases shouldn’t be automated. Edge cases are, by definition, usually one-off test cases, and the effort to automate edge cases generally does not pay off. Exploratory tests are best used to gain knowledge of a new feature and then tweak or revise tests based on the new knowledge” [279].

The article in [280] discussed some of the risks associated with automated tests such as: (1) the development team may rely too much on automation as the primary indicator of quality; (2) the team might disconnect with the Application and lose sight of the user; (3) applications can become less intuitive. They also presented cases where testing should not be handled by automation. For example, ad-hoc/exploratory testing, session based testing, vulnerability/security testing, and negative testing should not be automated because changing the data manually to reflect negative behavior and edge cases are best handled by manual testing due to the complexity that is often involved.

RonitSoen [281] stated, “The biggest problem with this great approach is the lack of documentation that leads to difficulties in reproducing defects and understanding the steps that led the tester to find these defects. Creativity, exploration, efficiency, all sound great, but defect remediation relies on the ability to understand the steps that the tester performed before running into the defect, which requires accurate and full documentation” [281].

Lisa Crispin criticized manual testing in her book, saying, “When manual testing is subjected to schedule pressure, the quality of the testing goes down. People begin to cut corners, omit tests, and miss problems. This is the kind of dysfunctional behavior for which traditional software development is famous. All those manual test cases look great
on paper, but when crunch time hits, all the paper goes out the window” [282].

On the other hand, Naidu said, “While automation continues to evolve rapidly, it is too early in the technological revolution to replace manual testing completely with automation. In fact, most of the new features, complex validations and business intensive functionalities will continue to be tested manually. The goal of 100% automation is not just ambitious, it is also impractical. Typically 70% automation helps maximize return on investments. Hence, manual testing remains and will continue to dominate in organizations with lower levels of automation maturity and in areas where ROI on automation is not significant [sic]” [283].

He also continued to highlight some important issues about automated testing: “Automation is not just a one-time initiative. Failure rates for automation are higher than success rates, and therefore careful planning is essential. Automation skills and tools are expensive” [283].

Johnson agreed, and wrote about the importance of skills in automated testing, “So, does automation replace the need for testers? No. Does automation change the role and skills required to fulfill the quality verification task? Yes” [284].

Manual testing can be very time consuming, but testers are free to explore and attempt to break the system, while that is not possible in automated testing as its goal is not to break the system but to notify when a change in the code broke a test.

As a result, having automated testing in an organization can be a somewhat controversial decision. Many automation efforts fail or do not meet the expected return on investment [284].
6.5.2 Uses of AHP in testing

From the previous review and various opinions regarding the automated testing, we noticed that there are several different factors that can play significant roles in the decision to have completely automated testing. These factors are: 1) Time constraint, 2) developers knowledge or understanding, 3) cost of test execution, 4) testing coverage, 5) resource availability, and 6) dealing with GUI applications. Therefore, AHP can be applied to help select one of the following testing techniques: automated testing, semi-automated, and manual testing.

Ranking the release indicators is another area where AHP can be beneficial to TDD. It tells the developers when they can say the software is ‘good enough’ for release or that testing might stop at this point. The National Institute of Standards and Technology provided a number of non-analytical methods and criteria to help decide when software is good enough to be released [285]. Therefore, to improve test efficiency, AHP can be introduced to evaluate the following indicators of deciding whether to release the product: 1) number of test cases completed, 2) percentage of defects reported, and 3) percentage of code executed without error. These indicators will be judged based on their impact on development cost, design, team velocity, code quality, and simplicity.

6.5.3 Automated testing decision

The first use of AHP is to decide the level of testing automation based on the provided criteria. The essential criteria that affect the decision will be discussed as follows.
6.5.3.1 Proposed criteria for the automation decision

To accomplish this goal, evaluators were asked to assess the testing options based on the following four main criteria:

- **Time**: which testing option is best in terms of spending less time in the testing phase?
- **Developer Understanding**: which testing option is easier for the developers to understand and apply?
- **Cost of execution**: which testing option costs less when executed?
- **Test coverage**: which testing option can cover more of the source code that has been tested?
- **Resource**: which testing option is best in terms of resource consumption?
- **GUI**: which testing option is best when dealing with GUI testing?

6.5.3.2 AHP-test decision structure

The AHP-Test Decision Structure includes three levels. The top level is the main objective, which is deciding the level of testing automation; the second level is the following criteria: time, developer understanding, cost of execution, test coverage, resources, and graphic user interface; and the third level is the options of automated testing, semi-automated testing, and manual testing. Figure 6.53 illustrates the AHP structure for the problem.
6.5.3.3 Pairwise comparison process for the automation decision

All participants were required to evaluate these testing alternatives based on certain criteria. For this purpose, sheets of paper with appropriate AHP tables were handed to all students in order to facilitate the comparison process.

The participants first compared the criteria using the Saaty scale from 1-9. The participants were asked questions as follows:

- Which is more important: time or developer understanding and by how much?
- Which is more important: time or cost of execution and by how much?
- Which is more important: time or test coverage and by how much?
- Which is more important: time or resource and by how much?
- Which is more important: time rogue testing and by how much?
- Which is more important: developer understanding or cost of execution and by how much?
- Which is more important: developer understanding or test coverage and by how much?
• Which is more important: developer understanding or resources and by how much?
• Which is more important: developer understanding rogue testing and by how much?
• Which is more important: cost of execution or test coverage and by how much?
• Which is more important: cost of execution or resources and by how much?
• Which is more important: cost of execution or GUI testing and by how much?
• Which is more important: test coverage or resources and by how much?
• Which is more important: test coverage or GUI testing and by how much?
• Which is more important: resources or GUI testing and by how much?

After finishing the criteria comparisons, the participants had to evaluate all testing techniques based on each criterion. Example follows:

• In term of the time, which testing techniques will save us more time and how much is that?

All the following comparisons were conducted based on each criterion:

(Automation X semi-automation), (automation X manually), (semi-automation X manually).

These questions and comparisons were repeated until the testing options were rated based on each criterion.
6.5.3.4 AHP evaluation results for the automation decision

1. Educational case studies

For Team 1, the rankings of the testing alternatives based on all criteria, i.e. time, developer understanding, cost of execution, test coverage, resources and GUI, are summarized as follows. First: Automation (38.59); Second: semi-automated (32.00); Third: manually (29.41). Table 6.102 summarizes the results.

Figure 6.54 shows the importance of each criterion as follows: code coverage (28.83), cost of execution (19.98), developer understanding (15.01), time (14.75), GUI (11.18) and resources (10.25).

The rankings of the testing alternatives by Team 2 are summarized as follows: First: Automation (44.92); Second: semi-automated (32.00); Third: manually (23.08). Table 6.103 summarizes the results.

Figure 6.55 shows the importance of each criterion as follows: cost of execution (34.68), time (20.53), code coverage (20.17), resources (11.19), developer understanding (7.04) and GUI (5.68).
6.5.3.5 Observations (educational case studies)

1. Considering the criteria as a whole, both teams selected automated testing to be the best option.

2. Team 1 considered code coverage to be the most important criteria, while Team 2 considered the cost of execution to be the most important criteria.

3. Testing manually was ranked as the last option for both teams.

4. If we consider each criterion individually, we can see that Team 1 ranked manual testing as the top option in three criteria: developer understanding, availability of resources, and GUI. Automated testing was ranked as the top option in the other three criteria, time, cost of execution, and code coverage. See table 6.104.

5. Team 2 ranked automated testing the highest in terms of time, cost of execution, availability of resources, and GUI. See table 6.105.
6. Team 2 ranked manual testing the highest in terms of developer understanding, and ranked semi-automated testing the highest in terms of GUI. See table 6.105.

Table 6.104 Automated testing level by Team 1 based on each criterion individually

<table>
<thead>
<tr>
<th>Technique</th>
<th>Time</th>
<th>Technique</th>
<th>Developer Understanding</th>
<th>Technique</th>
<th>Cost of Execution</th>
<th>Technique</th>
<th>Code Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automation</td>
<td>56.52 %</td>
<td>Manually</td>
<td>60.81 %</td>
<td>Automation</td>
<td>47.78 %</td>
<td>Automation</td>
<td>47.21 %</td>
</tr>
<tr>
<td>Semi-Automation</td>
<td>23.68 %</td>
<td>Semi-Automation</td>
<td>24.89 %</td>
<td>Semi-Automation</td>
<td>33.11 %</td>
<td>Semi-Automation</td>
<td>30.34 %</td>
</tr>
<tr>
<td>Manually</td>
<td>10.80 %</td>
<td>Automation</td>
<td>14.29 %</td>
<td>Manually</td>
<td>19.11 %</td>
<td>Manually</td>
<td>22.46 %</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Technique</th>
<th>Resources</th>
<th>Technique</th>
<th>GUI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manually</td>
<td>51.41 %</td>
<td>Manually</td>
<td>58.02 %</td>
</tr>
<tr>
<td>Semi-Automation</td>
<td>33.81 %</td>
<td>Semi-Automation</td>
<td>26.34 %</td>
</tr>
<tr>
<td>Automation</td>
<td>14.78 %</td>
<td>Automation</td>
<td>15.64 %</td>
</tr>
</tbody>
</table>

Table 6.105 Automated testing level by Team 2 based on each criterion individually

<table>
<thead>
<tr>
<th>Technique</th>
<th>Time</th>
<th>Technique</th>
<th>Developer Understanding</th>
<th>Technique</th>
<th>Cost of Execution</th>
<th>Technique</th>
<th>Code Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automation</td>
<td>62.00 %</td>
<td>Manually</td>
<td>69.71 %</td>
<td>Automation</td>
<td>45.51 %</td>
<td>Semi-Automation</td>
<td>39.52 %</td>
</tr>
<tr>
<td>Semi-Automation</td>
<td>28.27 %</td>
<td>Semi-Automation</td>
<td>20.24 %</td>
<td>Semi-Automation</td>
<td>32.00 %</td>
<td>Automation</td>
<td>31.36 %</td>
</tr>
<tr>
<td>Manually</td>
<td>9.73 %</td>
<td>Automation</td>
<td>10.05 %</td>
<td>Manually</td>
<td>22.50 %</td>
<td>Manually</td>
<td>29.12 %</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Technique</th>
<th>Resources</th>
<th>Technique</th>
<th>GUI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automation</td>
<td>65.14 %</td>
<td>Automation</td>
<td>50.75 %</td>
</tr>
<tr>
<td>Semi-Automation</td>
<td>26.06 %</td>
<td>Semi-Automation</td>
<td>29.04 %</td>
</tr>
<tr>
<td>Manually</td>
<td>8.80 %</td>
<td>Manually</td>
<td>20.21 %</td>
</tr>
</tbody>
</table>
2. Industrial case studies

The rankings of the testing alternatives by company A are summarized as follows: First: Automation (61.27); Second: semi-automated (24.94); Third: manually (13.79). Table 6.106 summarizes the results.

Figure 6.56 shows the importance of each criterion as follows: developer understanding (27.12), cost of execution (21.6), code coverage (16), resources (14.87), time (12.36) and GUI (8.06).

| Table 6.106 Level of automated testing by company A |
|-----------------|---------|
| Technique       | All     |
| Automation      | 61.27 % |
| Semi-Automation | 24.94 % |
| Manually        | 13.79 % |

The ranking for the testing alternatives by Company B is summarized as follows: First: Automation (45.09); Second: semi-automated (34.09); Third: manually (20.82). Table 6.107 summarizes the results.

Figure 6.57 shows the importance of each criterion as follows: code coverage (27.85), cost of execution (17.6), resources (16.33), developer understanding (15.16), time (11.55) and GUI (11.51).
Table 6.107 Level of automated testing by company B

<table>
<thead>
<tr>
<th>Technique</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automation</td>
<td>45.09 %</td>
</tr>
<tr>
<td>Semi-Automation</td>
<td>34.09 %</td>
</tr>
<tr>
<td>Manually</td>
<td>20.82 %</td>
</tr>
</tbody>
</table>

The rankings for the testing alternatives by company Care summarized as follows:

First: Automation (57.72); Second: semi-automated (28.66); Third: manually (13.62).

Table 6.108 summarizes the results.

Figure 6.58 shows the importance of each criterion as follows: developer understanding (27.85), resources (19.95), code coverage (17.41), time (15.18), cost of execution (11.77) and GUI (11.33).

Table 6.108 Level of automated testing by company C

<table>
<thead>
<tr>
<th>Technique</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automation</td>
<td>57.72 %</td>
</tr>
<tr>
<td>Semi-Automation</td>
<td>28.66 %</td>
</tr>
<tr>
<td>Manually</td>
<td>13.62 %</td>
</tr>
</tbody>
</table>

Figure 6.58 Importance of the criteria for company C in automated testing.
6.5.3.6 Observations (industrial case studies)

1. Considering the criteria as a whole, the three companies provided the same rankings: the highest rank was automated testing, the second semi-automated testing, and the last was manual testing.

2. Developer understanding was considered the most important criteria for company A and C, while company B considered code coverage to be the highest concern.

3. All three companies ranked the GUI criteria as the least important, showing that it has no significant impact on the decision.

4. If we consider each criterion individually, we can see that all three companies ranked automated testing in the top position for all criteria, with the exception of company B which ranked semi-automation the highest for the GUI criteria. See tables 6.109, 6.110, and 6.111.

5. It is very clear that manual testing was the least preferable option for the three companies.
### Table 6.109 Automated testing level by company (A) based on each criterion individually

<table>
<thead>
<tr>
<th>Technique</th>
<th>Time</th>
<th>Technique</th>
<th>Developer Understanding</th>
<th>Technique</th>
<th>Cost of Execution</th>
<th>Technique</th>
<th>Code Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automation</td>
<td>68.91%</td>
<td>Automation</td>
<td>63.19 %</td>
<td>Automation</td>
<td>62.21 %</td>
<td>Automation</td>
<td>57.22 %</td>
</tr>
<tr>
<td>Semi-Automation</td>
<td>22.09%</td>
<td>Semi-Automation</td>
<td>22.68 %</td>
<td>Semi-Automation</td>
<td>23.85 %</td>
<td>Semi-Automation</td>
<td>26.71 %</td>
</tr>
<tr>
<td>Manually</td>
<td>9 %</td>
<td>Manually</td>
<td>14.12 %</td>
<td>Manually</td>
<td>13.95 %</td>
<td>Manually</td>
<td>16.07 %</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Technique</th>
<th>Resources</th>
<th>Technique</th>
<th>GUI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automation</td>
<td>63.03 %</td>
<td>Automation</td>
<td>50.08 %</td>
</tr>
<tr>
<td>Semi-Automation</td>
<td>24.16 %</td>
<td>Semi-Automation</td>
<td>34.52 %</td>
</tr>
<tr>
<td>Manually</td>
<td>12.81 %</td>
<td>Manually</td>
<td>15.4 %</td>
</tr>
</tbody>
</table>

### Table 6.110 Automated testing level by company (B) based on each criterion individually

<table>
<thead>
<tr>
<th>Technique</th>
<th>Time</th>
<th>Technique</th>
<th>Developer Understanding</th>
<th>Technique</th>
<th>Cost of Execution</th>
<th>Technique</th>
<th>Code Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automation</td>
<td>64.14%</td>
<td>Automation</td>
<td>44.87 %</td>
<td>Automation</td>
<td>36.66 %</td>
<td>Automation</td>
<td>50.88 %</td>
</tr>
<tr>
<td>Semi-Automation</td>
<td>29.19%</td>
<td>Semi-Automation</td>
<td>33.98 %</td>
<td>Semi-Automation</td>
<td>35.81 %</td>
<td>Semi-Automation</td>
<td>31.94 %</td>
</tr>
<tr>
<td>Manually</td>
<td>6.67 %</td>
<td>Manually</td>
<td>21.16 %</td>
<td>Manually</td>
<td>27.53 %</td>
<td>Manually</td>
<td>17.18 %</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Technique</th>
<th>Resources</th>
<th>Technique</th>
<th>GUI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automation</td>
<td>50.79 %</td>
<td>Semi-Automation</td>
<td>40.71 %</td>
</tr>
<tr>
<td>Semi-Automation</td>
<td>32.20 %</td>
<td>Automation</td>
<td>29.84 %</td>
</tr>
<tr>
<td>Manually</td>
<td>17.00 %</td>
<td>Manually</td>
<td>29.45 %</td>
</tr>
</tbody>
</table>

### Table 6.111 Automated testing level by company (C) based on each criterion individually

<table>
<thead>
<tr>
<th>Technique</th>
<th>Time</th>
<th>Technique</th>
<th>Developer Understanding</th>
<th>Technique</th>
<th>Cost of Execution</th>
<th>Technique</th>
<th>Code Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automation</td>
<td>69.64%</td>
<td>Automation</td>
<td>51.74 %</td>
<td>Automation</td>
<td>45.59 %</td>
<td>Automation</td>
<td>62.85 %</td>
</tr>
<tr>
<td>Semi-Automation</td>
<td>23.42%</td>
<td>Semi-Automation</td>
<td>29.49 %</td>
<td>Semi-Automation</td>
<td>36.42 %</td>
<td>Semi-Automation</td>
<td>25.67 %</td>
</tr>
<tr>
<td>Manually</td>
<td>6.94 %</td>
<td>Manually</td>
<td>18.78 %</td>
<td>Manually</td>
<td>17.99 %</td>
<td>Manually</td>
<td>11.75 %</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Technique</th>
<th>Resources</th>
<th>Technique</th>
<th>GUI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automation</td>
<td>56.86 %</td>
<td>Automation</td>
<td>72.46 %</td>
</tr>
<tr>
<td>Semi-Automation</td>
<td>30.86 %</td>
<td>Semi-Automation</td>
<td>21.07 %</td>
</tr>
<tr>
<td>Manually</td>
<td>12.28 %</td>
<td>Manually</td>
<td>6.47 %</td>
</tr>
</tbody>
</table>
6.5.4 Secondly: Ranking the release indicators

The second use of AHP is to rank the software release indicators. The release indicators and the proposed criteria will be discussed as follows.

6.5.4.1 Proposed criteria for release indicators

To accomplish this goal, evaluators were asked to evaluate release indicators based on the following four main criteria:

- **Cost**: which release indicator costs more (money and effort)?
- **Design**: which release indicator will enhance the design more?
- **Velocity**: which release indicator will speed up the team velocity?
- **Code quality**: which release indicator will improve code quality more?
- **Simplicity**: which release indicator is simpler to do and finish?

6.5.4.2 Release indicators

In this section, we will rank the three important release indicators proposed by the National Institute of Standards in order to help decision makers assess whether software is good enough to be released [285]. These indicators are: (1) Number of test cases completed; (2) percentage of defects reported; and (3) percentage of code executed without error. They will be evaluated based on the criteria proposed in this section.
6.5.4.3 AHP-release indicators structure

The AHP-Release Indicators Structure includes three levels. The top level is the main objective of finding the most important release indicators; the second level is the following criteria: cost, design, velocity, code quality, and simplicity; and the third level is the options of test coverage, unit test run, percentage of defects found, and percentage of execution without error. Figure 6.59 illustrates the AHP structure for the problem.

![Figure 6.59 AHP structure for release indicators ranking](image)

6.5.4.4 Pairwise comparison process for release indicators

Sheets of paper with the appropriate AHP tables were handed to all participants in order to facilitate the process of the comparison.

The participants first compared the criteria using the Saaty scale from 1-9. The participants were asked questions as follows:

- Which is more important: cost or design and by how much?
- Which is more important: cost or velocity and by how much?
- Which is more important: cost or code quality and by how much?
- Which is more important: cost or simplicity and by how much?
- Which is more important: design or velocity and by how much?
- Which is more important: design or code quality and by how much?
• Which is more important: design or simplicity and by how much?
• Which is more important: velocity or code quality and by how much?
• Which is more important: velocity or simplicity and by how much?
• Which is more important: code quality or simplicity and by how much?

After finishing the criteria comparisons, the participants had to evaluate all the release indicators based on each criterion, exactly how it was previously shown in the automated testing decision.

6.5.4.5 AHP evaluation results for the release indicators

1. Educational case studies

For Team 1, the rankings of the release indicators based on all criteria, i.e. cost, design, velocity and code quality, and simplicity, are summarized as follows. First: percentage of execution without error (37.79); Second: percentage of defects found (34.89); Third: percentage of test cases completed (27.33). Table 6.112 summarizes the results.

Figure 6.60 shows the importance of each criterion as follows: code quality (26.87), cost (21.40), simplicity (20.14), design (18.33), and velocity (13.25).

<table>
<thead>
<tr>
<th>Release Indicators</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>% Of Execution Without Error</td>
<td>37.79 %</td>
</tr>
<tr>
<td>% Of Defects Reported</td>
<td>34.89 %</td>
</tr>
<tr>
<td>% Of Test Cases Completed</td>
<td>27.33 %</td>
</tr>
</tbody>
</table>

Table 6.112 Release indicators ranking by Team 1

Figure 6.60 Importance of the criteria for Team 1 in release indicators.
The rankings of the release indicators by Team 2 are summarized as follows: First: percentage of test cases completed (38.55); Second: percentage of defects found (35.67); Third: percentage of execution without error (25.78). Table 6.113 summarizes the results.

Figure 6.61 shows the importance of each criterion as follows: cost (43.87), velocity (20.49), code quality (18.29), design (9.92), and simplicity (7.44).

<table>
<thead>
<tr>
<th>Release Indicators</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>% Of Test Cases Completed</td>
<td>38.55 %</td>
</tr>
<tr>
<td>% Of Defects Reported</td>
<td>35.67 %</td>
</tr>
<tr>
<td>% Of Execution Without Error</td>
<td>25.78 %</td>
</tr>
</tbody>
</table>

Table 6.113 Release indicators ranking by Team 2

Figure 6.61 Importance of the criteria for Team 2 in release indicators.
6.5.4.6 Observations (educational case studies)

1. Considering the criteria as a whole, Team 1 and Team 2 had opposing results. Team 1 ranked percentage of execution without error in the highest position, while Team 2 ranked it in the lowest position. On the contrary, Team 2 ranked percentage of test cases competed in the highest rank, while Team 1 ranked it in the lowest position. Both team ranked percentage of defects in the second position.

2. Team 1 considered code quality to be the most important criteria, while Team 2 considered cost to be the most important criteria.

3. If we consider the rankings based on each criterion individually, we can see that Team 1 ranked percentage of execution at the top in three criteria: cost, code quality and simplicity. In addition, they ranked percentage of test cases completed at the top in terms of design criteria, and percentage of defects reported at the top in terms of velocity. See table 6.114.

4. Team 2 ranked percentage of test cases completed the highest in terms of cost, design, and velocity, and ranked the percentage of the defect indicator the highest in terms of code quality and simplicity. See table 6.115.
**Table 6.114 Release indicators by Team 1 based on each criterion individually**

<table>
<thead>
<tr>
<th>Release Indicators</th>
<th>Cost</th>
<th>Release Indicators</th>
<th>Design</th>
<th>Release Indicators</th>
<th>Velocity</th>
<th>Release Indicators</th>
<th>Code Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>% Of Test Cases Completed</td>
<td>46.38 %</td>
<td>% Of Test Cases Completed</td>
<td>41.48 %</td>
<td>% Of Defects Reported</td>
<td>48.1 %</td>
<td>% Of Execution Without Error</td>
<td>38.83 %</td>
</tr>
<tr>
<td>% Of Defects Reported</td>
<td>38.64 %</td>
<td>% Of Defects Reported</td>
<td>31.99 %</td>
<td>% Of Execution Without Error</td>
<td>29.77 %</td>
<td>% Of Defects Reported</td>
<td>32.16 %</td>
</tr>
<tr>
<td>% Of Test Cases Completed</td>
<td>18.98 %</td>
<td>% Of Execution Without Error</td>
<td>26.53 %</td>
<td>% Of Test Cases Completed</td>
<td>22.13 %</td>
<td>% Of Test Cases Completed</td>
<td>29 %</td>
</tr>
</tbody>
</table>

**Table 6.115 Release indicators by Team 2 based on each criterion individually**

<table>
<thead>
<tr>
<th>Release Indicators</th>
<th>Cost</th>
<th>Release Indicators</th>
<th>Design</th>
<th>Release Indicators</th>
<th>Velocity</th>
<th>Release Indicators</th>
<th>Code Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>% Of Test Cases Completed</td>
<td>39.67 %</td>
<td>% Of Test Cases Completed</td>
<td>54.93 %</td>
<td>% Of Defects Reported</td>
<td>39.22 %</td>
<td>% Of Defects Reported</td>
<td>41.68 %</td>
</tr>
<tr>
<td>% Of Defects Reported</td>
<td>31.21 %</td>
<td>% Of Defects Reported</td>
<td>32.21 %</td>
<td>% Of Defects Reported</td>
<td>38.27 %</td>
<td>% Of Test Cases Completed</td>
<td>31.17 %</td>
</tr>
<tr>
<td>% Of Execution Without Error</td>
<td>29.11 %</td>
<td>% Of Execution Without Error</td>
<td>12.87 %</td>
<td>% Of Execution Without Error</td>
<td>22.51 %</td>
<td>% Of Execution Without Error</td>
<td>27.15 %</td>
</tr>
</tbody>
</table>

**Release Indicators**
- % Of Execution Without Error
- % Of Defects Reported
- % Of Test Cases Completed

**Simplicity**
- % Of Execution Without Error
- % Of Defects Reported
- % Of Test Cases Completed

---

294
2. Industrial case studies

The rankings for release indicators by company A are summarized as follows: First: percentage of execution without error (47.13); Second: percentage of defects found (31.06); Third: percentage of test cases completed (21.8). Table 6.116 summarizes the results.

Figure 6.62 shows the importance of each criterion as follows: design (33.59), code quality (22.66), cost (19.31), velocity (13.21), and simplicity (11.24).

<table>
<thead>
<tr>
<th>Release Indicators</th>
<th>All</th>
<th>% Of Execution Without Error</th>
<th>47.13 %</th>
<th>% Of Defects Reported</th>
<th>31.06 %</th>
<th>% Of Test Cases Completed</th>
<th>21.8 %</th>
</tr>
</thead>
</table>

The rankings for release indicators by company B are summarized as follows: First: percentage of defects found (36.07); Second: percentage of test cases completed (33.29); Third: percentage of execution without error (30.64); Table 6.117 summarizes the results.

Figure 6.63 shows the importance of each criterion as follows: design (31.42), code quality (19.9), velocity (18.19), cost (17.57), and simplicity (13.25).
The rankings for release indicators by company Care summarized as follows: First: percentage of defects found (37.96); Second: percentage of execution without error (36.13); Third: percentage of test case completed (25.19). Table 6.118 summarizes the results.

Figure 6.63 shows the importance of each criterion as follows: code quality (25.81), simplicity (24.51), design (23.48), velocity (13.83), and cost (12.36).

Table 6.117 Release indicators ranking by company B

<table>
<thead>
<tr>
<th>Release Indicators</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>% Of Defects Reported</td>
<td>36.07 %</td>
</tr>
<tr>
<td>% Of Test Cases Completed</td>
<td>33.29 %</td>
</tr>
<tr>
<td>% Of Execution Without Error</td>
<td>30.64 %</td>
</tr>
</tbody>
</table>

Figure 6.63 Importance of the criteria for company B in release indicators.

Table 6.118 Release indicators ranking by company C

<table>
<thead>
<tr>
<th>Release Indicators</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>% Of Defects Reported</td>
<td>37.96 %</td>
</tr>
<tr>
<td>% Of Execution Without Error</td>
<td>36.13 %</td>
</tr>
<tr>
<td>% Of Test Cases Completed</td>
<td>25.19 %</td>
</tr>
</tbody>
</table>

Figure 6.64 Importance of the criteria for company C in release indicators.
6.5.4.7 Observations (industrial case studies)

1. Considering the criteria as a whole, company B and company C ranked percentage of defects reported as the highest, while company A ranked percentage of execution without error as the highest.

2. Design and code quality was considered the most important criteria for companies A and B, while company C considered code quality and simplicity to be the highest concerns.

3. If we consider each criterion individually, we can see that company A ranked percentage of execution without error in the highest position based on all criteria except design, in which case percentage of defects reported was the highest. See table 6.119.

4. Company B ranked percentage of defects at the top in terms of cost, velocity and simplicity. They ranked percentage of test cases completed at the top in terms of design, and percentage of execution without error at the top in terms of code quality. See table 6.120.

5. Company C ranked percentage of defects reported the highest based on all criteria except design, in which case percentage of execution without error was the highest. See table 6.121.
### Table 6.119 Release indicators by company (A) based on each criterion individually

<table>
<thead>
<tr>
<th>Release Indicators</th>
<th>Cost</th>
<th>Release Indicators</th>
<th>Design</th>
<th>Release Indicators</th>
<th>Velocity</th>
<th>Release Indicators</th>
<th>Code Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>% Of Execution Without Error</td>
<td>49.61 %</td>
<td>% Of Defects Reported</td>
<td>39.94 %</td>
<td>% Of Execution Without Error</td>
<td>45.67 %</td>
<td>% Of Defects Reported</td>
<td>61.84 %</td>
</tr>
<tr>
<td>% Of Test Cases Completed</td>
<td>28.08 %</td>
<td>% Of Execution Without Error</td>
<td>39 %</td>
<td>% Of Test Cases Completed</td>
<td>29.08 %</td>
<td>% Of Test Cases Completed</td>
<td>20.03 %</td>
</tr>
<tr>
<td>% Of Defects Reported</td>
<td>22.31 %</td>
<td>% Of Test Cases Completed</td>
<td>21.06 %</td>
<td>% Of Defects Reported</td>
<td>25.25 %</td>
<td>% Of Test Cases Completed</td>
<td>18.93 %</td>
</tr>
</tbody>
</table>

- **Release Indicators (Simplicity)**
  - % Of Execution Without Error: 52.42 %
  - % Of Defects Reported: 30.53 %
  - % Of Test Cases Completed: 17.05 %

### Table 6.120 Release indicators by company (B) based on each criterion individually.

<table>
<thead>
<tr>
<th>Release Indicators</th>
<th>Cost</th>
<th>Release Indicators</th>
<th>Design</th>
<th>Release Indicators</th>
<th>Velocity</th>
<th>Release Indicators</th>
<th>Code Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>% Of Defects Reported</td>
<td>39.41 %</td>
<td>% Of Test Cases Completed</td>
<td>39.59 %</td>
<td>% Of Defects Reported</td>
<td>39.94 %</td>
<td>% Of Test Cases Completed</td>
<td>48.55 %</td>
</tr>
<tr>
<td>% Of Test Cases Completed</td>
<td>34.88 %</td>
<td>% Of Defects Reported</td>
<td>32.96 %</td>
<td>% Of Test Cases Completed</td>
<td>27.45 %</td>
<td>% Of Defects Reported</td>
<td>32.29 %</td>
</tr>
<tr>
<td>% Of Execution Without Error</td>
<td>25.71 %</td>
<td>% Of Execution Without Error</td>
<td>27.45 %</td>
<td>% Of Execution Without Error</td>
<td>21.20 %</td>
<td>% Of Test Cases Completed</td>
<td>19.16 %</td>
</tr>
</tbody>
</table>

- **Release Indicators (Simplicity)**
  - % Of Defects Reported: 38.38 %
  - % Of Execution Without Error: 35.84 %
  - % Of Test Cases Completed: 25.33 %

### Table 6.121 Release indicators by company (C) based on each criterion individually

<table>
<thead>
<tr>
<th>Release Indicators</th>
<th>Cost</th>
<th>Release Indicators</th>
<th>Design</th>
<th>Release Indicators</th>
<th>Velocity</th>
<th>Release Indicators</th>
<th>Code Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>% Of Defects Reported</td>
<td>34.29 %</td>
<td>% Of Execution Without Error</td>
<td>36.54 %</td>
<td>% Of Defects Reported</td>
<td>39.71 %</td>
<td>% Of Defects Reported</td>
<td>44.00 %</td>
</tr>
<tr>
<td>% Of Execution Without Error</td>
<td>33.89 %</td>
<td>% Of Test Cases Completed</td>
<td>31.97 %</td>
<td>% Of Execution Without Error</td>
<td>38.00 %</td>
<td>% Of Test Cases Completed</td>
<td>35.12 %</td>
</tr>
<tr>
<td>% Of Test Cases Completed</td>
<td>31.82 %</td>
<td>% Of Defects Reported</td>
<td>31.56 %</td>
<td>% Of Test Cases Completed</td>
<td>22.29 %</td>
<td>% Of Test Cases Completed</td>
<td>20.89 %</td>
</tr>
</tbody>
</table>

- **Release Indicators (Simplicity)**
  - % Of Defects Reported: 40.04 %
  - % Of Execution Without Error: 36.93 %
  - % Of Test Cases Completed: 23.02 %
6.6 Analysis and Discussion

This section presents the feedback obtained from the participants regarding the use of AHP in XP development. First, the answers for the semi-interview are presented. Second, the questionnaires results are summarized. Third, the collected data and research questions are linked with the propositions. Finally, the criteria for the interpretation of the results are applied.

6.6.1 Semi-structured interview

The semi-structured interview was conducted after showing the participants the results of the AHP evaluation for all the XP practices. Some of the results were surprising and others were expected. The interview included open questions to obtain students’ general opinions about the AHP, the advantages and disadvantage of using the AHP, and the best experiences with the AHP among all XP practices. As said previously, the data was collected in the form of handwritten notes during the interviews. These notes were organized in a folder to be analyzed and reached easily. The questions and answers for the semi-interview are presented below. All participants’ names were kept anonymous:

1. What do you like about the AHP?

   • “AHP helps in making decisions, especially for large projects with lots of requirements. It provides techniques to prioritize the requirements and release management”.

   • “I like the AHP because it makes difficult decisions so easy”.

   • “It is a sophisticated procedure to choose between various weighted outputs. And the results of AHP are very close to reality. So when we do this process, we are pretty much sure that, what we do is correct and that’s a moral boost”.
• “It is a good process that helps us make the best decision. Particularly, it helps us make good decisions suitable to the task”.

• “If AHP is used correctly it will assist the decision makers in standardizing the decision making process”.

• “AHP has the ability to provide analytical assessment at several levels of detail”.

• “AHP is a practical tool for making decisions throughout a project’s development. It is a very good practice to consider all team members’ opinions when making a decision, but if every decision requires a vote from all team members, then the decision-making process will be slow. Here is where AHP can be best suit XP development”.

• “The AHP satisfies everyone or at least reduces conflicts in the team. If one of the team members does not listen to his/her colleagues’ ideas and always thinks that his/her ideas are the best, this will lead to bottlenecks between the team members and they will not be able to complete their project successfully. But when using AHP it was very rare that we had conflicting opinions”.

• “The idea of ‘do not focus on what you may need for tomorrow’ had its own certain drawbacks. Because the complete requirements were provided at the beginning of the project, from time to time certain functionalities were neglected, despite it being obvious that we would need them for tomorrow. This also led to a few cases where some user stories stretched from one iteration to the next because additional functionalities should have been added to them later. In my opinion, simplicity cane expensive in terms of redoing things. AHP can reduce the effort and risks when making such a decision due to its ability to consider all the important factors affecting the team decision”.

• “AHP is helpful in inspiring courage. For example when insisting on tackling the problem until it was solved, courage for the refactoring waste best experience I had”.

• “It is an easy model to express the ideas in an analysis”.

• “AHP provides a way of finding solutions that best suit one’s goals and actually increases understanding of the problem, instead of just finding the ‘correct’
decision. This is very true in many applications where decision makers only tend to prescribe one reasonable answer”.

2. Where do you think AHP was the most beneficial (in which Practice)? Why?

• “AHP was useful in the planning game phase of XP programming”.

• “AHP is most important in pair programming. For example, when we did pairing for the advanced software process class, we could match two people and choose the correct partner. Since I am a novice in programming, I learnt a lot from my teammates”.

• “In the ‘planning game’ task because in XP, we have to make a decision every week (iteration) based on the user stories we receive. So, sometimes the task is much more complicated and we cannot come to the best conclusion. Applying AHP, will help us make the best decision for each iteration and yet also reduce our time spent”.

• “I think AHP is beneficial to prioritize the user stories and any conflicting team member decisions”.

• “In formalization of the decision making process I prefer it more in planning game and refactoring”.

• “AHP’s ability in providing analytical assessment for user story sequencing. This in most cases helps reduce test cases and consequently helps reduce the use of the team’s resources, eventually reducing the project’s cost”.

• “In my opinion, AHP was very good in the design stage. Team members should realize the concepts and convert them into applicable relationships between different elements of the ‘to be released product’ for the current iteration. Any miscommunication will end up in an incomplete design which cannot meet the requirements. AHP was used as a roadmap for the team to use the simplest tool and prioritize the CRC cards. Without AHP, I am not sure how we would have selected the simplest design tool or prioritize the CRC cards. Also, I would say that it is also good for refactoring. I found it difficult at the beginning, but when I became familiar with it, it became so easy and beneficial”.

• “Sometimes people are stubborn and don’t want to follow the team’s consensus. This can cause trouble for all team members and for the outcome of the product.
If this happens and we cannot control the direction of the project, one of the suggestions was to listen to those with more experience to justify and encourage the understanding of everyone on the team; however, this idea means ignoring the voices of the other team members and relying only on one person. I think that eventually this will conflict with the ultimate idea of XP. Therefore, AHP is needed to compromise between opinions in a formalized way, even though it takes some time to get the final AHP results.”

• “Pair programming is a social skill that takes time to learn. AHP is very good to help select the best pairs. My partner had more experience and helped me a lot to learn new things. In the beginning I felt a bit shy because I was asking many questions, but he was understanding and did not get bothered. We were striving to cooperate, which requires give and take from both partners regardless of corporate status. AHP also was good in refactoring; I had to focus only on the eight refactoring techniques were proposed in this project”.

• “In pair programming because it can perfectly match the best programmers”.

• “It provides a comprehensive and rational framework for structuring a decision problem, for representing and quantifying its elements, for relating those elements to overall goals, and for evaluating alternative solutions. Generally, for instance, in the first step, it determines the relative weights of the decision criteria. Secondly, it determines the relative priority of alternatives”.

• “In all of the XP areas we tried there were benefits, but refactoring benefitted more than others”.

3. What are the advantages you found in using AHP with XP?

• “It is simple and helpful in decision-making”.

• “I would say that it makes pairwise comparisons straightforward and convenient”.

• “Like I said, the exchange of knowledge is the major advantage”.

• “First, it helps solve conflicts and make the best decisions for the task. Second, it enables fast processing. By this I mean AHP can help us reach a conclusion for the task faster”.

• “The strong consensus”.
• "It provides me with several weighted scenarios for my project approach work streams. This helps me as a PM to consider mitigation risk suppression through the project approach”.

• "The most beneficial part of AHP is maximizing the communication level among the project members and creating positive discussion. It conveys the message, information and sometimes even knowledge; when teams or team members do not communicate to convey these things, it can produce an imbalance in the project, the project may be stalled, no further progress can be made, and may even prove to be fatal”.

• "With AHP, teams can effectively accomplish their work more accurately and more efficiently. Discussions not only update people on the progress of the project but also inspire the members. Using AHP brings everyone’s voice to the development, and ensures that no one is superior”.

• "Investigating the best criteria, resulting in a clean code and helping to make the best decisions”.

• "It is based on a mathematical base which strengthens this tool and proves its power”.

• "AHP reduces bias in decision making by checking the consistency of the evaluation measures and alternatives”.

• "It supports group decision-making through consensus by calculating the geometric mean of the individual pairwise comparisons”.

4. What are the disadvantages you found in using AHP with XP?

• "The number of comparisons sometimes is too high”.

• "Some important factors may be missed since when doing agile development, unexpected factors appear from nowhere and the team has to be able to restructure AHP with added factors if they were not counted. What I want to say here is that when structuring AHP we need to consider the real factors, which our experience in this level might not help us do”.

• "AHP requires the collaboration of all project members as well as customers."
Basically, AHP needs to consider all of the project’s members’ opinions in order to evaluate the factors, problems and make a good decision. It has to be all of them in order to come to the best conclusion”.

• “It is time consuming, so the question is whether it’s worth it or not. It depends on the project and the team decision. There is no general answer can apply to all XP projects”.

• “AHP’s limitation is in cross-correlating different projects’ work scenarios”.

• “AHP might be more beneficial for experts and not junior programmers. When practicing XP, there were two considerable gaps in programming skills in educational environments compared to real business. First, there is the gap between the project requirements and average programming skills of each team. Second was the difference between the programming skills of the individuals”.

• “There are a lot of advantages when using AHP in XP development. It creates energy when working in a group and shares ownership and knowledge transfer. We improved our social skills and improved overall programming skills”.

• “It is difficult to implement a project with such great discussion. For example, if we need to provide the design within 2 hours, the team cannot discuss it for more than 20 minutes (since we need time for coding). How can this problem be resolved? I don’t have a clear answer, and I cannot say that AHP is the best for all XP practices, because it does take time”.

• “It is difficult to distinguish among the weights using the 9-point scale”.

• “The variety of results because it’s based on personal opinions”.

• “It is time-consuming. Secondly, the information can be lost by a complete aggregation method of the additive type”.

• “The artificial nature of the 9-point scale is another disadvantage”.

5. If you were a decision maker in XP development, would you use AHP?

• “Yes I would use it. I always prefer to make decisions with proper reasoning and AHP is one such kind of tool. However, it may also depend on the project size,
complexity and available time.”

• “Yes, I would recommend it”.

• “I would use AHP when important elements of the decision are difficult to quantify or compare, or where communication among team members is impeded by their different specializations, terminologies, or perspectives”.

• “Yes, I’d use the AHP because it would help me reduce the time required and make the best decision based on the importance of the task”.

• “I would”.

All the remaining answers are “Yes”, without comments.

6. Do you have any ideas or suggestions for using AHP with XP?

• “I am currently doing co-op work for SaskPower and in our project we are using an agile methodology and partially implementing AHP techniques. I observed that we were able to build a quality product, but the project completion time increased by approximately 20% (because of continued interaction with clients and constant changes in requirements)”.

• “Maybe a mobile AHP tool?”

• “Doing some of the cost and risk analysis for using AHP in XP development”.

• “Adding the ability to cross-correlate work streams in order to help the PM to mitigate the resource shortages during the project’s execution”.

• “Applying AHP to another area to see how it works and observing the results”.

• “Improving the way of the calculating”.

Others didn’t have any suggestions in mind.
Overall, AHP received very positive feedback from the participants. They found that AHP resolved conflicting opinions and brought every team member’s voice to the decision in a practical way. It also inspires courage among the team by letting every opinion be heard. Planning game, simple design, pair programming, and refactoring were seen as benefiting from AHP. However, refactoring was seen as the XP practice which most benefited from AHP. The time and the number of the comparisons, and the 9-point scale were the main concerns of the participants. All of them recommended using AHP in the future with XP. A few recommendations were proposed, such as developing an automated tool to reduce the time for AHP calculation, adding mobility features, doing cost and risk analysis, and trying it with other XP areas and studying the outcome.

6.6.2 Questionnaires

The questionnaires given to the participants were aimed at obtaining their perceptions and experiences with AHP. The questionnaires were divided into two main parts. The first part considered the questions about AHP as a decision and ranking tool as follows:

1. Evaluate AHP as a decision tool used in XP from two aspects: decision quality and practicality.

The second part considered questions regarding the direct benefit to the XP practices and investigated the participants’ satisfaction.

2. Evaluate AHP’s impact on team and user satisfaction for each XP practice.

The details of these questions can be seen in Appendix3.

We used a seven-point Likert scale to reflect the level of acceptability of AHP. The following depicts the meanings of the seven-point scale:
1. Totally unacceptable
2. Unacceptable.
3. Slightly unacceptable.
5. Slightly acceptable.
6. Acceptable.

When the participants finished the questionnaires, we calculated the results. We presented the total percentage of the acceptability for each statement in tables 4, 5, and 6. The total percentage of the acceptability was calculated as follows:

- The total percentage of acceptability (TPA) =

  \[(\text{the average of the score for each team} \times 100)/7.\]

- The average of the score for each team =

  the sum of the scores given by the team members/number of team members.

The following tables show the acceptability level for AHP as a tool for ranking and decision making, as well as the acceptability level of AHP in each XP practice. The results were obtained from the two educational cases and the three industrial cases. Note: Figures are rounded down to the nearest integer, as any additional digits are an artifact of the calculation rather than an indication of higher precision. The main purpose of showing the percentage is to show the level of AHP acceptability to the users.
1. *AHP acceptability level as a decision and ranking tool*

In table 6.122, we can clearly see all the results are positive. However, the lowest percentages were given by Team 1 (59%) and Team 2 (62%), and regarded time efficiency.

<table>
<thead>
<tr>
<th>AHP as a decision tool used in Extreme Programming (Team 1, Team 2)</th>
<th>Team 1</th>
<th>Team 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A- Decision Quality</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Capturing the needed Information</td>
<td>76%</td>
<td>88%</td>
</tr>
<tr>
<td>Clarity of the decision process</td>
<td>88%</td>
<td>86%</td>
</tr>
<tr>
<td>Clarity of criteria involved</td>
<td>81%</td>
<td>76%</td>
</tr>
<tr>
<td>Clarity of the alternatives involved</td>
<td>81%</td>
<td>79%</td>
</tr>
<tr>
<td>Goodness of the decision structure</td>
<td>86%</td>
<td>90%</td>
</tr>
<tr>
<td><strong>B- Practically</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Understandability</td>
<td>83%</td>
<td>88%</td>
</tr>
<tr>
<td>Simplicity</td>
<td>71%</td>
<td>86%</td>
</tr>
<tr>
<td>Time Efficiency</td>
<td>59%</td>
<td>62%</td>
</tr>
<tr>
<td>Reliability</td>
<td>74%</td>
<td>76%</td>
</tr>
</tbody>
</table>
2. Acceptability level of AHP ranked by Team 1

Overall, AHP’s impacts on Team 1 were positive. However, they gave the lowest percentages of (47%) and (48%) regarding the satisfaction of the final results of AHP and the satisfaction of the impacts of AHP on the prioritization tools. The highest and lowest percentages in each practice are highlighted.

Table 6.123 Acceptability level of AHP ranked by Team 1

<table>
<thead>
<tr>
<th>Team 1</th>
<th>Prioritization Tools</th>
<th>Simple Design Tools</th>
<th>CRC Cards</th>
<th>Best Pair</th>
<th>Rules in Pairing</th>
<th>Refactoring</th>
<th>Automated Testing</th>
<th>Release Indicators</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A- Team Impact</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Improvement in the Team communication</td>
<td>83%</td>
<td>93%</td>
<td>80%</td>
<td>83%</td>
<td>74 %</td>
<td>76%</td>
<td>76%</td>
<td></td>
</tr>
<tr>
<td>Creating an informative discussion and learning opportunity</td>
<td>74 %</td>
<td>90%</td>
<td>83%</td>
<td>86%</td>
<td>79%</td>
<td>71%</td>
<td>74 %</td>
<td>71%</td>
</tr>
<tr>
<td>Clarifying the ranking problem</td>
<td>86%</td>
<td>86%</td>
<td>93%</td>
<td>83%</td>
<td>90%</td>
<td>71%</td>
<td>81%</td>
<td>78%</td>
</tr>
<tr>
<td>Resolving the conflict opinions among the members.</td>
<td>78%</td>
<td>78%</td>
<td>74%</td>
<td>78%</td>
<td>74%</td>
<td>64%</td>
<td>76%</td>
<td>76%</td>
</tr>
<tr>
<td>Elevating the performance of the team</td>
<td>74%</td>
<td>86%</td>
<td>83%</td>
<td>79%</td>
<td>83%</td>
<td>76%</td>
<td>81 %</td>
<td>79 %</td>
</tr>
<tr>
<td><strong>B- Satisfaction</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Satisfaction of the final results of AHP</td>
<td>47%</td>
<td>86%</td>
<td>88 %</td>
<td>64%</td>
<td>86%</td>
<td>76%</td>
<td>83%</td>
<td>71%</td>
</tr>
<tr>
<td>Satisfaction of the impacts on the XP practice</td>
<td>48%</td>
<td>88 %</td>
<td>80%</td>
<td>62%</td>
<td>80%</td>
<td>74%</td>
<td>86%</td>
<td>76%</td>
</tr>
</tbody>
</table>
3. Acceptability level of AHP ranked by Team 2

AHP’s impacts on Team 2 were mainly positive. However, the lowest percentage given by Team 2 (45%) dealt with the satisfaction of the impacts on the prioritization tools. The highest and lowest percentages in each practice are highlighted.

Table 6.124 Acceptability level of AHP ranked by Team 2

<table>
<thead>
<tr>
<th>Team 2</th>
<th>Prioritization Tools</th>
<th>Simple Design Tools</th>
<th>CRC Cards</th>
<th>Best pair</th>
<th>Rules in pairing</th>
<th>Refactoring</th>
<th>Automated testing</th>
<th>Release indicators</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>A-Team Impact</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Improvement in the Team communication</td>
<td>86%</td>
<td>88%</td>
<td>86%</td>
<td>90%</td>
<td>94%</td>
<td>93%</td>
<td>94%</td>
<td>88 %</td>
</tr>
<tr>
<td>Creating an informative discussion and learning opportunity</td>
<td>93%</td>
<td>93%</td>
<td>90%</td>
<td>90%</td>
<td>93%</td>
<td>88%</td>
<td>90%</td>
<td>88 %</td>
</tr>
<tr>
<td>Clarify the ranking problem</td>
<td>93%</td>
<td>94%</td>
<td>93%</td>
<td>93%</td>
<td>90%</td>
<td>90%</td>
<td>88 %</td>
<td>79 %</td>
</tr>
<tr>
<td>Resolving the conflict opinions among the members.</td>
<td>93%</td>
<td>88 %</td>
<td>88 %</td>
<td>93%</td>
<td>93%</td>
<td>86%</td>
<td>86%</td>
<td>83%</td>
</tr>
<tr>
<td>Elevating the performance of the team</td>
<td>88 %</td>
<td>93%</td>
<td>90%</td>
<td>94%</td>
<td>90%</td>
<td>88%</td>
<td>90%</td>
<td>86%</td>
</tr>
<tr>
<td><strong>B- Satisfaction</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Satisfaction of the final results of AHP</td>
<td>80%</td>
<td>94%</td>
<td>83%</td>
<td>83%</td>
<td>93%</td>
<td>88%</td>
<td>93%</td>
<td>86%</td>
</tr>
<tr>
<td>Satisfaction of the impacts on the XP practice.</td>
<td>45%</td>
<td>88 %</td>
<td>79%</td>
<td>86%</td>
<td>94%</td>
<td>93%</td>
<td>94%</td>
<td>88 %</td>
</tr>
</tbody>
</table>
4. Acceptability level of AHP ranked by the three companies

AHP received positive ratings by the three companies on most of the questions. However, the lowest percentage given by the three companies (61%, 57%, 57%) had to do with time efficiency. The highest and lowest percentages in each practice are highlighted.

Table 6.125 Acceptability level of AHP ranked by the three companies

<table>
<thead>
<tr>
<th></th>
<th>Company A</th>
<th>Company B</th>
<th>Company C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1- AHP as a Ranking tool in Refactoring</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>A- Decision Quality:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Capturing the needed information</td>
<td>86%</td>
<td>83%</td>
<td>88%</td>
</tr>
<tr>
<td>Clarity of the decision process</td>
<td>94%</td>
<td>94%</td>
<td>90%</td>
</tr>
<tr>
<td>Clarity of the criteria involved</td>
<td>88%</td>
<td>83%</td>
<td>88%</td>
</tr>
<tr>
<td>Clarity of the Alternatives involved</td>
<td>88%</td>
<td>90%</td>
<td>95%</td>
</tr>
<tr>
<td>Goodness of the decision structure</td>
<td>98%</td>
<td>88%</td>
<td>71%</td>
</tr>
<tr>
<td><strong>B- Practically</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Understandability</td>
<td>98%</td>
<td>90%</td>
<td>85.66</td>
</tr>
<tr>
<td>Simplicity</td>
<td>76%</td>
<td>74%</td>
<td>74%</td>
</tr>
<tr>
<td>Time Efficiency</td>
<td>61%</td>
<td>57%</td>
<td>57%</td>
</tr>
<tr>
<td>Reliability</td>
<td>81%</td>
<td>90%</td>
<td>90%</td>
</tr>
<tr>
<td>2- Direct Benefit into the refactoring</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>A- Team Impact</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Improvement in team communication</td>
<td>93%</td>
<td>93%</td>
<td>94%</td>
</tr>
<tr>
<td>Creating an informative discussion and learning opportunity</td>
<td>86%</td>
<td>95%</td>
<td>93%</td>
</tr>
<tr>
<td>Clarifying the ranking problem</td>
<td>86%</td>
<td>93%</td>
<td>86%</td>
</tr>
<tr>
<td>Resolving the conflict opinions among the members.</td>
<td>86%</td>
<td>93%</td>
<td>83%</td>
</tr>
<tr>
<td>Elevating the performance of the team</td>
<td>79%</td>
<td>86%</td>
<td>Not study</td>
</tr>
<tr>
<td><strong>B- Satisfaction</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Satisfaction of the final results of AHP.</td>
<td>88%</td>
<td>90%</td>
<td>83%</td>
</tr>
<tr>
<td>Satisfaction of the impacts on the refactoring.</td>
<td>88%</td>
<td>90%</td>
<td>Not study</td>
</tr>
</tbody>
</table>
6.6.3 Logic linking of the data to the research questions and propositions

In this section, we will provide the logic linking of the collected data to the main research questions and the initial study propositions. The table below shows how the collected data and the study propositions are connected.

<table>
<thead>
<tr>
<th>Propositions</th>
<th>Collected data from the assigned project</th>
<th>Questions in the semi-structured interview</th>
<th>Questions in the questionnaire</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sections 6.1, 6.2, 6.3, 6.4, and 6.5.</td>
<td>1, 2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1, 2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>2, 3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>2, 3</td>
<td>2</td>
</tr>
</tbody>
</table>

6.6.4 Interpretation of results

The collected data obtained from multiple data sources will be analyzed by comparing it with the propositions according to the criteria for interpretation mentioned in section 5.2. Accordingly, the answers to the study propositions are as follows:

*P1: AHP captures important criteria and alternatives that need to be considered in each adopted XP practice. Also, AHP’s results show the order of importance for each alternative.*

- The AHP structure can be used for the following: (1) prioritizing user stories from the customer’s perspective; (2) prioritizing user stories from the developer’s perspective; (3) ranking the prioritization techniques; (4) selecting the design tool; (5) ranking the CRC cards for Team 1; (6) ranking the CRC cards for Team 2; (7) selecting the pairs; (8) ranking the rules in pair matching; (9) ranking the refactoring techniques based on the internal attributes; (10) evaluating the refactoring techniques based on the external attributes; (11) automated testing
decisions; (12) and release indicators ranking showed that the criteria and alternatives are considered and well structured.

All the AHP structures can be seen in Figures 6.15, 6.16, 6.19, 6.22, 6.25, 6.27, 6.29, 6.35, 6.41, 6.47, 6.53, and 6.59.

In each XP practice which used AHP, the results and goals accomplished are as follows:

- Table 6.25 shows the developers’ ranking of the user stories, and the highest ranked result was user story 3. Table 6.26 shows the customers’ ranking of the user stories and the highest ranked result was user story 3 as well.
- The prioritization techniques were ranked in tables 6.28 and 6.29 by Team 1 and Team 2, and the results were the relative weighting tool for Team 1 and Team 2.
- For selecting the design tools, tables 6.32 and 6.33 showed the whiteboard tool was selected by Team 1 and Team 2.
- CRC cards were ranked in tables 6.40 and 6.46 and the results were: the Issue Priority Class card for Team 1; and the UserForm Class card for Team 2.
- In pair programming, the pair options were ranked in tables 6.48 6.49, 6.52, 6.53 and 6.53. The result was Expert-Expert as it was ranked the highest by all participants: Team 1, Team 2, Company A, Company B, and Company C.
- Tables 6.58, 6.59, 6.62, 6.63, and 6.64 showed the ranking for the rules of pairing two persons. The results were: almost identical for Team 1, marginally different for Team 2, and almost identical for companies A, B, and C.
- Tables 6.68, 6.69, 6.72, 6.73, and 6.74 showed the ranking for the refactoring techniques based on the internal attributes. The results were: Extract Class for
Team 1, Inline Class for Team 2, Extract Method for Company A, and Extract Class for companies B and C.

- Similarly, the ranking for the refactoring techniques based on the external attributes can be seen in tables 6.78, 6.79, 6.82, 6.83, and 6.84. The results were: Extract Class for Team 1 and Team 2, Inline Class for Company A, and Extract Class for companies A and B.

- For the automated testing decision, tables 6.102, 6.103, 6.106, 6.107, and 6.107 showed that the result ranked highest by all participants was automated testing.

- Finally, the ranking for the release indicators can be seen in tables 6.112, 6.113, 6.116, 6.117, and 6.118. The results were: execution without error for Team 1, percentage of test cases completed for Team 2, percentage of execution without error for Company A, percentage of defects found for Company B, and percentage of execution without error for Company C.

P2: AHP facilitates the process of selection, ranking, prioritization and decision-making when it is needed in the XP practice.

- Table 6.122 showed the acceptability level of AHP as a decision tool by Team 1 and Team 2. AHP received good feedback in terms of decision process (88% and 86%, respectively), goodness of the decision structure (86% and 90%, respectively), in terms of understandability (83% and 88%, respectively), simplicity (71% and 86%, respectively), and reliability (74% and 76%, respectively).
• In addition, table 6.125 showed the acceptability level of AHP as a decision tool as judged by companies, B, and C. AHP received good feedback in terms of the decision process (94%, 94% and 90%, respectively), goodness of the decision structure (98%, 88% and 71%, respectively), understandability (98%, 95% and 85%, respectively), simplicity (76%, 74% and 74%, respectively), and reliability (81%, 90% and 90%, respectively).

P3: AHP involves an informative discussion and improves communication among developers.

• In table 6.123, Team 1 supported this statement for the following uses of AHP: prioritization tools (74%), simple design tools (90%), CRC cards (83%), best pairs (86%), rules in pairing (79%), refactoring (71%), automated testing (74%), and release indicators (71%).

• In table 6.124, Team 2 supported this statement for the following uses of AHP: prioritization tools (93%), simple design tools (93%), CRC cards (90%), best pairs (90%), rules in pairing (93%), refactoring (88%), automated testing (90%), and release indicators (88%).

• In addition, the three companies provided their opinions on the advantages of creating informative discussion among team members. Table 6.125 shows the results as follows: Company A (86%), Company B (95%) and Company C (93%).

• The semi-structured interviews presented in section 6.6.1 indicated positive answers to questions 1, 2 and 3 that supported this proposition.
P4: *AHP resolves conflicting opinions among the XP team when applying a specific practice.*

- In table 6.123, Team 1 agreed that AHP resolved the conflicting opinions in the following uses of AHP with XP practices: prioritization tools (78%), simple design tools (78%), CRC cards (74%), best pairs (78%), rules in pairing (74%), refactoring (64%), automated testing (76%), and release indicators (76%).

- In table 6.124, the results for Team 2’s judgments for AHP’s use in resolving conflicting opinions are as follows: prioritization tools (93%), simple design tools (88%), CRC cards (88%), best pairs (93%), rules of pairing (93%), refactoring (86%), automated testing (86%), and release indicators (83%).

- Companies A, B and C provided their opinions on the advantage of resolving the conflicting opinions in table 6.125. The results in order are 86%, 93% and 83%, respectively.

- The semi-structured interviews presented in section 6.6.1 indicated positive answers to questions 1, 2 and 3 that supported this proposition.
6.7 Validity

Construct validity, internal validity, external validity and reliability describe common threats to the validity tony performed study [123].”Empirical studies in general and case studies in particular are prone to biases and validity threats that make it difficult to control the quality of the study to generalize its results” [286]. In this section, relevant validity threats are described. A number of possible threats to validity can be identified for this work.

Parts of this section were also published in The Eighth International Conference on Software Engineering Advances (CSEA 2013), Italy, October 27- November 1, 2013.

6.7.1 Construct validity

Construct validity deals with the correct operational measures for the concept being studied and researched. The major construct validity threat to this study is the small number of participants in each case study. This threat was mitigated by using several techniques in order to ensure the validity of the findings, as outlined below.

- Data triangulation: A major strength of case studies is the possibility of using many different sources of evidence [126]. This issue was taken into account through the use of surveys and interviews with different types of participants from different environments with various levels of skills and experience, and through the use of several observations as well as feedback from those involved in the study. By establishing a chain of evidence, we were able to reach a valid conclusion.
• Methodological triangulation: The research methods employed were a combination of a project conducted to serve this purpose, interviews, surveys, AHP result comparisons, and researchers’ notes and observations.

• Member checking: Presenting the results to the people involved in the study is always recommended, especially for qualitative research. This was done by showing the final results to all participants to ensure the accuracy of what was stated and to guard against researcher bias.

6.7.2 Internal validity

Internal validity is only a concern for an explanatory case study [126]. Internal validity focuses on establishing a causal relationship between students and educational constraints. This issue can be addressed by relating the research questions to the propositions and other data sources providing information regarding the questions.

6.7.3 External validity

External validity is related to the domain of the study and the possibilities of generalizing the results. To provide external validity to this study, we will need to conduct an additional case study in the industry involving experts and developers, and then observe the similarities and the differences in the findings of both studies. Thus, future work will contribute to accruing external validity.
6.7.4 Reliability

Reliability deals with the data collection procedure and results. Other researchers should arrive at the same case study findings and conclusions if they follow the same procedure. We addressed this by making the research questions, case study set up, data collection and analysis procedure plan is available for use by other researchers.
CHAPTER 7

CONCLUSION AND FUTURE WORK

In this chapter, conclusions and suggestions for future research work are presented. AHP and its potential uses in XP development are summarized in Section 7.1. Section 7.2 discusses possible areas for future work.

7.1 Summary

This thesis focused on providing the XP team with a framework based on AHP that can be injected into several XP practices for the purpose of solving decision or ranking problems. The contributions of this thesis to 10 areas of XP practices involve the benefits of applying the AHP technique. These contributions, identified in Chapter 1, are repeated here with further justification.

The Analytic Hierarchy Process (AHP) is a very well established and structured technique that can capture and analyze various tangible and intangible attributes of very complex multiple criteria decision-making problems. XP teams encounter, on a daily basis, various situations that need a decision, prioritization of factors or a case-based evaluation to maximize the effectiveness of the software being developed. AHP appears to be an important technique to create a cooperative decision-making environment that accelerates development with XP and reconciles the differences in opinion between team members. Special attention needs to be placed on the computational needs of AHP to ensure its cost-effectiveness. This work has presented many potential XP areas used AHP along with results from the educational and industrial environment.
The XP planning game practice was the first practice into which we introduced AHP, as can be seen in section 6.1. Two potential areas were investigated: prioritizing the user stories and ranking the prioritization techniques. After using AHP to investigate the critical factors that influence the decisions involved in prioritizing the user stories from the perspectives of both the customers and the developers, it was found that AHP was an important tool for reconciling the conflicting opinions between the developers and customers on one side, and the other stakeholders on the other. The idea that the developer played a role in prioritizing the user stories with the customers was instrumental in determining the right priorities which serve the different goals of both parties. When the developers informed the customers about the impacts of the risk, cost, and complexity associated with each user story, they were able to determine the order of implementing each one. In fact, not only did the customer receive more information and clarity about the project through developer inputs, but also the developers themselves benefited from participating in the prioritization activity, by increasing the communication and positive discussion with the customers.

The second use was to rank the common prioritization techniques used in XP development to prioritize the user stories. AHP provided a very good vision for developers when they wanted to decide which prioritization technique was the most preferable. Considering factors of simplicity, time, accuracy and collaboration when selecting the tools provided many advantages to the XP team, including the stakeholders. In our cases, the relative weighting technique was the most preferable, but the method we chose was general and it should be noted that the ranking can change depending on the team. More importantly, though, AHP helped students evaluate each prioritization
technique from different viewpoints. In addition, they could mathematically reconcile the conflicting opinions among them. However, under time pressure, the teams in our case studies failed to use the relative weighting tools as they were ranked the highest by both teams. Both teams decided to use the tool which was ranked the highest based on the time criteria: MoScoW for one team and the Top-Ten for the other.

In chapter 6.2, we also explored using AHP in XP simple design practice. The first use of AHP was to select the simplest design tool for the XP team. The design tools were evaluated based on four criteria: simplicity, communication, documentation, and portability. The whiteboard tool was ranked the highest. They participants found that AHP accelerated the design phase and created an informative discussion. The second use of AHP was to prioritize the CRC cards as a common simple design tool. It was found that AHP helped provide a good vision for the XP team to understand which CRC cards have the most impact on the system. Considering responsibility, collaboration, and stability when prioritizing CRC cards brought many advantages to the XP team such as measuring the quality of the software, implementing the most valuable class to design and ensuring the simplicity of the design.

In pair programming, discussed in chapter 6.3, AHP was used for two purposes. First was to select the best matching pairs based on different goals such as speed, learning, sharing knowledge, and code quality considering four alternatives: expert-expert, expert-average, expert-novice and novice-novice. It was found that the expert-expert pairing was the best matching in most of the studies. Another use of AHP was to decide if the pairs should have the same characteristics or can be different. The selection was based on the same factors mentioned previously and three selections were proposed: almost identical,
marginally different, and clearly different.

In the refactoring, discussed in chapter 6.4, AHP was used to rank the refactoring techniques based on the internal and external quality attributes. We chose four internal quality attributes to be the core criteria for the ranking of AHP refactoring methods: cohesion, coupling, complexity, and code size. Additionally, four external quality attributes were chosen to be the main criteria for the refactoring pattern evaluation: reusability, flexibility, maintainability and understandability. It was found and validated that the two refactoring patterns Extract Class and Extract Method were mostly ranked in the top positions on both the internal and external quality attributes. Moreover, the developers significantly reduced the time usually spent for refactoring after using AHP with the proposed refactoring patterns.

In the testing discussed in chapter 6.5, AHP was used to help make decisions about the level of automated testing. Such decisions can be affected by factors such time constraints, developer knowledge and understanding, cost of test execution, testing coverage, availability of resources, and dealing with GUI applications. Therefore, AHP was applied to help select one of the testing techniques: automated testing, semi-automated testing, and manual testing. The best result by participant consensus was automated testing.

Another use for AHP in the TDD is to rank the software release indicators, such as number of test cases completed, percentage of defects reported, and percentage of code executed without error. These indicators are evaluated based on cost, design, velocity, code quality and simplicity. The results varied due to the differences in the projects conditions such as criteria and alternatives, project size, time and budget.
Overall, in the XP environment, AHP received positive feedback from all participants in terms of the quality of the decision, the team performance, and user satisfaction level. Nevertheless, some XP areas benefited more than others, like refactoring. AHP can improve team performance in all their decisions leading to reducing the duration of some XP practices such as refactoring. It also helps the XP team avoid many conflicting opinions while encouraging every member to express his or her thoughts. The best scenario for using AHP occurs when the decision hinges on independent criteria and alternatives, and when the number of the comparisons is not both large and complicated. One more thing has to be made clear regarding the use of AHP: time. Time appeared to be an issue in some of the XP practices, specifically in planning game, as indicated in tables 6.123, 6.124 and 6.125. The research has several limitations as follows:

- The criteria have been used are not mutually exclusive due to the limitation of the AHP technique. Some of these criteria can be defined by the decision makers in the organization of the customers.
- The criteria adopted in the research are not mutually exclusive; this is a limitation of the AHP technique. Some of the criteria can be defined by the decision makers in the customers’ organization.
- The large projects running for a long-time and involving many teams are not considered because of the communications issues attached to this type of projects. In fact, large projects and large teams are currently a topic of debate regarding the applicability of XP, and not only AHP.
• The criteria were defined based on the project that was developed and the customer requirements. Each project might have different requirements and different customers needs which inevitably leads to change some of these criteria.
• AHP is developed to work with a small number of criteria and alternatives which meet our needs in XP. However, for a larger number of objectives, the ANP might be the best choice.

Moreover, adding more factors or more people to the study doesn’t mean that the results will necessarily improve, but it means we might need to integrate AHP with another tool, or use the Analytical Network Process (ANP) which works with more sophisticated problems. However, if this the case, then the idea of the simplicity in XP will not likely be achieved.

7.2 Future Work

1. Further areas in XP to be investigated

There are more areas in XP which can benefit from using AHP. Notably, during planning game practice, AHP can be used to select the appropriate technique for user story estimation. There are four techniques defined in [287]: expert opinion, analogy, disaggregation, and planning Poker. The estimation techniques can be evaluated based on four criteria: simplicity, time, team collaboration, and accuracy. The team can follow the results of AHP, which takes everyone’s opinion from different angles based on different criteria.

Also, there is potential for using AHP with the XP metaphor. The system metaphor is one XP practice that has been lately defined and poorly understood. XP team members
can present many metaphors and AHP can be used to select the best metaphor. It is very important to be consistent when explaining the system to avoid conflict among the team members. Considering various criteria before approving a metaphor can elevate team productivity. There are five criteria very essential for evaluating any proposed metaphors: 1) system components coverage, 2) expandability, 3) correctness, and 4) vocabulary.

In XP testing, there are some potential areas as well. For example, AHP can be used to select the best test case candidates to be automated based on: 1) availability of the test data, 2) clarity of the execution steps, 3) repeatability of the test execution, and 4) frequency of use by the user. Another example is to use AHP to evaluate different available automated testing tools with strong influencing factors such as: 1) compatibility, 2) functionality, and 3) flexibility, affordability, and maintainability. In fact, several difficulties are encountered when deciding to use automating testing tools. As it has been said, “Buying a wrong tool is listed as the no one challenging test automation because no matter what kind of process or organization you have, if the tool is not a good technical or business fit, people will not be able to apply it” [288].

2. AHP-XP responsive tool

Developing an automated AHP tool that is designed to meet the XP values and its development environment will make AHP even more beneficial. It should also consider the time issue which arose in some of the XP practices and quickly respond to the changes and the requests of the recalculation.
3. **Cost and benefit analysis**

Each project has its own target, goal, and conditions that are different from other projects. So, deciding whether to apply the AHP decision tool or not can rely on the results of calculating and comparing the benefits and the costs the tool will bring to the project (cost and benefit analysis).

4. **Further quantitative study**

AHP has proved its ability in various areas in XP development. However, there is still a need for deeper investigation about AHP’s impact on the time, cost, and risk. This investigation should be done using a real industrial project in order to provide the most accurate results.
LIST OF REFERENCES


329


[60] Young Lee and Eum Hee Choi, “Applying the AHP Techniques to Electronic Commerce in a Special Attention to Fashion Website Selection”, in proceedings of the First International Conference on The Human Society and the Internet - Internet Related Socio-Economic, 2001, pp. 49-60.


332
[127] Wilson Ng, Aidan Gill, “Hypotheses versus Propositions”:


[228] Liming Zhao, Jane Huffman Hayes “Predicating Classes in Need of Refactoring: An Application of Static Metrics”, Department of Computer Science, University of Kentuck, 2006.


