



Norwegian University of
Science and Technology

Myrmidia

The Warhammer Fantasy Battle Army Builder

Glenn Rune Strandbråten

Master of Science in Computer Science

Submission date: June 2011

Supervisor: Anders Kofod-Petersen, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

Warhammer fantasy battle and 40k are two strategic games using miniature figures, where each of two players control one army. One of the challenges is constructing a “good” army, based on the rules of the games. Choices made as to the nature of the troops, their equipment, and so forth is typically a function of opponents and player style.

With the numerous races and units available in the Warhammer universe, may the task of selecting a good lineup become difficult and much experimentation may be required in order to create a “good” army. This project aims to develop a decision support system for army building for the Warhammer fantasy battle game. The system will, by relying on prior experiences and CBR help the player create their army; be able to explain the army composition based on the rules applied and why the prior knowledge were used.

Assignment given: 13. January 2011

Supervisor: Anders Kofod-Petersen

Abstract

In this thesis, I present an approach to a case-based reasoning system with explanation capabilities in the Warhammer Fantasy Battle domain. This product is meant to support Warhammer gamers in their initial army lineup, by providing suggestions based on previously successful games against an opposing horde. Explanations will be used in order to convey the reasoning behind the solution, to present the data the solution is based upon and why certain changes were made.

The created product is capable of creating the army lineup and give partially satisfactory explanations, based on the goals set both for the application as a whole and explanations. Although a full domain model is not implemented, are the results promising; with the inclusion of more domain knowledge and cases, will a fully competent and accurate system be achievable.

Keywords: Artificial Intelligence, Case-base reasoning, Explanation aware computing, Warhammer Fantasy Battle

Preface

This report constitutes my master thesis, which has been written and developed during the 10th semester of my Master of Science studies in Computer Science at the Norwegian University of Science and Technology (NTNU). The thesis is a continuation of the work done in the course TDT4500 – Intelligent Systems, Specialization Project. This work were performed at the department of Computer and Information Science (IDI), group for Artificial Intelligence between January 20th and June 9th, 2011.

I would first like to thank my supervisor Anders Kofod-Petersen. His invaluable feedback, knowledge and insight helped improve the quality of my work.

Secondly I would like to thank WarTrond for their time and effort in providing me with the initial data for the case-base.

Thirdly a thanks to my fellow students at ITV-363-Ugle for the discussions, talks and much needed breaks during the semester.

Finally a thanks to my family and friends for all the support they have given me during my education.

Trondheim, June 8, 2011
Glenn Rune Strandbråten

Contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 1 |
| 1.1 | Motivation and Background | 1 |
| 1.2 | Goals | 2 |
| 1.3 | Structure of the Thesis | 3 |
| 2 | Warhammer | 5 |
| 2.1 | Races and Units | 5 |
| 2.2 | Army creation | 8 |
| 3 | Design | 11 |
| 3.1 | Domain model | 11 |
| 3.1.1 | Phase 1 | 11 |
| 3.1.2 | Phase 2 | 13 |
| 3.1.3 | Phase 3 | 14 |
| 3.2 | Fundamental design | 17 |
| 3.2.1 | Case structure | 17 |
| 3.3 | Technologies used | 19 |
| 3.3.1 | jColibri | 19 |
| 3.3.2 | Apache Derby | 20 |
| 3.3.3 | Hibernate | 21 |
| 4 | Implementation | 25 |
| 4.1 | CBR | 25 |
| 4.1.1 | CBR framework | 26 |
| 4.1.2 | Similarity (Retrieve) | 28 |
| 4.1.3 | Adaptation (Reuse) | 31 |
| 4.1.4 | Revise and Retain | 36 |
| 4.2 | Explanations | 38 |
| 4.2.1 | Explanation framework | 40 |
| 4.3 | Warhammer and case representation | 43 |

| | | |
|----------|---|-----------|
| 5 | Testing and results | 45 |
| 5.1 | Unit weight similarity | 45 |
| 5.2 | Case adaption | 46 |
| 5.3 | Race exchange | 47 |
| 5.4 | Generated explanations | 47 |
| 6 | Discussion | 49 |
| 6.1 | Case-based reasoning | 49 |
| 6.1.1 | The case-base | 50 |
| 6.1.2 | Retrieve | 50 |
| 6.1.3 | Reuse | 51 |
| 6.1.4 | Revise | 52 |
| 6.1.5 | Retain | 53 |
| 6.2 | Explanation | 53 |
| 6.3 | Evaluation of goals | 54 |
| 6.3.1 | G1: Determine explanation goals | 54 |
| 6.3.2 | G2: Create the domain model | 55 |
| 6.3.3 | G3: System creation | 55 |
| 6.3.4 | G4: Evaluation | 55 |
| 6.4 | Future work | 56 |
| 7 | Conclusion | 59 |
| | Bibliography | 61 |
| A | Glossary | 65 |
| B | Test results | 67 |
| B.1 | Unit weights | 67 |
| B.2 | Adaption result | 71 |
| B.3 | Race exchange result | 72 |
| B.4 | Explanation prints | 73 |
| C | Cases | 79 |

List of Figures

- 2.1 A Warhammer Dwarf Lord unit 8

- 3.1 Phase 1 database schema 12
- 3.2 Phase 2 database schema 13
- 3.3 Phase 3 database schema 16
- 3.4 The create query UI 19
- 3.5 Embedded Derby Architecture 21
- 3.6 An example class for Hibernate mapping 22

- 4.1 The CBR cycle 26
- 4.2 CBR class diagram 27
- 4.3 Similarity weight configuration UI 30
- 4.4 The retrieval result UI 31
- 4.5 The revise UI 37
- 4.6 The retain UI 38
- 4.7 The CBR knowledge containers 39
- 4.8 Explanation class diagram 42
- 4.9 Warhammer and case class diagram 44

List of Tables

| | | |
|-----|---|----|
| 2.1 | Dwarf lord & Beastmen Chaos Spawn unit characteristics. | 7 |
| 2.2 | Army creation summary | 9 |
| 3.1 | Example of a simple case query | 17 |
| 3.2 | Example of a more extensive case query | 18 |
| 4.1 | Unit type similarity lookup table | 35 |
| 4.2 | Army type similarity lookup table | 35 |
| 4.3 | Weapon type similarity lookup table | 36 |
| B.1 | Unit similarity weight, Test: 1 | 67 |
| B.2 | Unit similarity weight, Test: 2 | 68 |
| B.3 | Unit similarity test: Empire | 69 |
| B.4 | Unit similarity test: Dwarfs | 70 |
| B.5 | Unit similarity test: High Elves | 71 |
| B.6 | Adaption result - unit preference error | 72 |
| B.7 | Race unit exchange table | 73 |
| C.1 | The 10 cases in the case-base | 79 |
| C.2 | Case army A1 | 80 |
| C.3 | Case army A2 | 81 |
| C.4 | Case army A3 | 81 |
| C.5 | Case army A4 | 82 |
| C.6 | Case army A5 | 83 |
| C.7 | Case army A6 | 84 |
| C.8 | Case army A7 | 85 |
| C.9 | Case army A8 | 86 |

Chapter 1

Introduction

This work is the continuation of my specialization project (Strandbråten, 2010), where I studied different: artificial intelligence approaches; explanation aware computing; existing technology, frameworks and libraries; existing products within the problem domain; and the Warhammer Fantasy Battle domain. The goal of the preliminary project was to get an overview of the existing products, technology and to create a rough outline for the architecture to be used in this master thesis project.

Case-Based Reasoning (CBR) is an emerging field within Artificial Intelligence, with many application areas and several projects are created with CBR at its core. Briefly put is CBR using knowledge about previously recorded events, to help solve newly arisen problems. The knowledge is stored as cases which consists of the problem description, the problem solution and its effectiveness. Some examples of CBR applications are: CASEY (Koton, 1988), CARMA (Branting et al., 1999), JULIA (Hinrichs, 1992), CREEK (Aamodt, 1994) and AmICREEK (Kofod-Petersen and Aamodt, 2009).

Explanation-aware computing is the act of supplying applications with enough knowledge to be able to explain its reasoning and thought processes to the user. By supplying such data, can the application help raise the users confidence in the results provided by the application. The concept of explanation-aware computing is covered in e.g.: Leake (1995a), Sørmo et al. (2005), Roth-Berghofer (2004); Roth-Berghofer and Cassens (2005) and Doyle et al. (2003). And examples of explanation-aware applications: ACCEPTER (Leake, 1995b) and SWALE (Kass and Leake, 1988).

The rest of this chapter gives the motivation and background for the work performed in this thesis in Section 1.1; the goals of the thesis is described in Section 1.2; and the chapter is concluded with the structure of the thesis in Section 1.3.

1.1 Motivation and Background

Since early in high school, when I was lured into a Dungeons & Dragons (D&D)¹ session for the first time by close friends, the game and concept have intrigued me. The con-

¹Dungeons and dragons official site: <http://www.wizards.com/DND/>

cept of venturing into a fantasy realm with a fictional character, battling monsters and generally have a lot of fun with friends is great. While we never had any serious games and the rules were bent or broken more often than not, they still formed the basis of our fun.

While Warhammer² was a known phenomena and to some extent very similar to D&D it was never brought up as a gaming system to try. When this assignment was offered at the university as a master thesis assignment I saw it as a great opportunity to get to learn the game, and at the same time work on something that I have a great interest for. During the discussions with the supervisor that offered the assignment; we talked about the possibility to change it into a D&D project, but this was discarded as the problem area is different; i.e.: In D&D you create one player; keep and develop that player for years. In Warhammer you must create and manage an entire army, where the army often is unique to the imminent battle, and as such is it better suited as an artificial intelligence task.

This task is suited as an artificial intelligence task and more precisely a CBR system, because of the almost unlimited number of possible unit and equipment permutations available; even when constricted by the rule set. Conventional statistics calculations becomes unmanageable when faced with a close to limitless supply of permutations. If you could remove all the permutations which technically is within the rule set, but gamers would find unsound or unmanageable, are the remaining number of permutations staggering. This is made even more complex by not having any legal limit on the number of points which can be used in a battle, resulting in exponentially more permutations when the points increase.

Technically this project offers challenges with respect to domain modeling, interpretation of the game rules and the explanation depth. Explanation aware computing is an area within computer science which is unknown to me; and as such will this project be a great introduction into this field. Furthermore as this problem domain resembles that of scheduling and planning in AI, links between these concepts may be drawn and evaluated, as a means to a possible problem solution.

1.2 Goals

The goals of this project are:

- G1** Determine which explanation goals the system should support.
- G2** Determine what information the system require to function, and how best to represent the domain model.
- G3** Create the system.
- G4** Evaluate the results (system).

²Warhammer Fantasy Battle official site: <http://www.games-workshop.com/gws/>

1.3 Structure of the Thesis

This thesis is meant to be read as a whole from beginning to end, and later chapters may refer to or build upon previous chapters. The remainder of this thesis is organized into six chapters.

In Chapter 2 the reader is introduced to Warhammer Fantasy Battle along with the most prominent rules for creating an army. Chapter 3 details the evolution of the domain model; the fundamental design and case structure; as well as the technologies used in the system. The implementation of the systems three major components (CBR, explanation and Warhammer- case representation) are described in Chapter 4. In Chapter 5 will the testing and their subsequent results be presented, while Chapter 6 features discussions about various aspects of the process. Including choices made towards the functionality in the four CBR steps and in in regards to the explanation capabilities; evaluation of the goals set forth in 1.2; and future work. Chapter 7 concludes the thesis.

A glossary containing all abbreviations and terminology used in this document can be found in Appendix A. The tables containing the test results is located in Appendix B and all cases represented in the system is located in Appendix C.

Chapter 2

Warhammer

Warhammer Fantasy Battle is a miniature figure board game, where each player leads an army to battle against another player. There are several races to choose from and each race have many different and unique units that can be added to the army. Each unit in Warhammer can have multiple different miniatures; some with or without optional equipment. These miniatures are made in plastic/iron on a 25mm scale and each units sits upon a square/rectangular socket. The miniature figures is delivered unpainted and sometimes unassembled, in official tournaments and games with hard core gamers are unpainted miniatures illegal and cannot be used. In these tournaments/games, must a player have the exact miniatures for his/hers setup, if his/hers spearmen have shields, then he/she must use points to equip those shields during the army creation and vice versa. These rules are more commonly relaxed in private matches or in less formal settings, although unpainted units are frowned upon. By supplying the units in an unpainted form, can each and every player bring something unique to his/hers army, by choosing the exact color scheme to be used on all the units; and this also appeals to the collector aspect of the game.

Strict rules govern the game play and army creation process. On the battlefield the individual units strength and abilities are matched against the opponent, and in conjunction with the lay of the land and the roll of the dice determine the winner. This essentially mean that the same two armies can do battle several times and the results will most likely be different each time.

2.1 Races and Units

There are 15 races to choose from and each race have a host of different units that can be bought and added to the players collection; additionally one unit may have different equipment from another unit of the same race and type. The average number of units available to a race are $\frac{437}{15} \approx 29$. Each unit is differentiated from the others based on it's *classification* and *characteristics*.

Classification determines what type of unit it is, and is divided into the following

categories (Games Workshop, 2009, p.80-87 & 134):

- *Cavalry* (Ca) units are man-sized riders on war beasts, usually warhorses or similar creatures. They specialize in deadly charges into the enemy units.
- *Chariots* (Ch) are wheeled war vehicles drawn by beasts and crewed with armed warriors.
- *Infantry* (In) units are all the human sized foot troops in the game, regardless of race. They normally represents the core of your army.
- *Monsters* (Mo) are the larges units on the battlefield and amongst the most powerful units in the game. Dragons and Greater Daemons are examples of monsters.
- *Monstrous Beasts* (MB) are lesser than monsters, but still formidable foes, like the Great Eagle.
- *Monstrous Cavalry* (MC) are much the same as ordinary cavalry, but tougher, and the riders uses monsters as mounts instead of horses.
- *Monstrous Infantry* (MI) are man-shaped units which is two or three times the size of normal infantry, but still fight on foot. This includes Trolls, Ogres and Minotaurs.
- *Swarms* (Sw) are magically summoned hordes of small creatures like rats, snakes or insects, their numbers compensate for their individual weakness.
- *Unique Units* (Un) are bizarre mechanical constructs or magical monstrosities that require a unique rule set by themselves.
- *War Beasts* (WB) are hunting animals like dogs or wolves which cover distances quickly and are dangerous in close combat.
- *War Machines* (WM) are powerful units which can be used to fight whole regiments of units or breach walls. The Empire Great Cannon is an example of a war machine.

Additionally some units are either a lord or a hero, these are powerful units which leads the army to war. They are not separated in their own classification, but can belong to any classification (usually infantry with the ability to purchase a mount). *Lords* are the most powerful unit in an army, with fearsome martial- or magical might. *Heroes* are lesser in might than lords, but still worth a score of ordinary warriors.

Characteristics describe all the aspects of a unit through nine associated attributes. Each attribute have a numeric value in the range from 0 to 10, all these attributes are uniquely described in a table for each unit and associated classification¹. Some units have a random attribute value which is determined by a dice roll and will be depicted in

¹An attribute value zero can be represented in the table as both a “0” and a “-”.

the table as: e.g.: 2D6, this describes that you should roll two six-sided dice to determine the value, when needed in battle. This random value may or may not be added to a constant attribute value. The following list is a description of the nine characteristics (Games Workshop, 2009, p.3):

- *Attacks* (A) relates to how many times the unit can attack in this round.
- *Ballistic skill* (BS) reflects how accurate the unit is with its ranged weapons.
- *Initiative* (I) is how fast the unit reacts, and reflects the order in which the units strike in close combat.
- *Leadership* (Ld) reflects how courageous, determined and self-controlled a unit is.
- *Movement Allowance* (M) determines the distance, in inches, a unit can move on the battlefield under normal circumstances.
- *Strength* (S) determines the strength of the unit. It is related to how hard the unit hits with its weapon, and the damage dealt in close combat.
- *Toughness* (T) is the units ability to resist physical damage and pain.
- *Wounds* (W) is the amount of damage that a unit can take before it is incapacitated or killed.
- *Weapon skill* (WS) determines how skilled a unit is with its weapon or how vicious the beast is.

An example of two unit characteristics is presented in Table 2.1.

Table 2.1: Dwarf lord & Beastmen Chaos Spawn unit characteristics.

| Unit | M | WS | BS | S | T | W | I | A | Ld | Type |
|-------------|-----|----|----|---|---|---|---|------|----|------|
| Dwarf lord | 3 | 7 | 4 | 4 | 5 | 3 | 4 | 4 | 10 | In |
| Chaos Spawn | 2D6 | 3 | 0 | 4 | 5 | 3 | 2 | D6+1 | 10 | MB |

The Dwarf lord depicted in Figure 2.1 have basically the same stats as mentioned in the characteristic above, but may have additional information based on the equipment he carries. As a lord this unit is capable of leading the entire army to war upon the battlefield.



Figure 2.1: A Warhammer Dwarf Lord unit (Image: Anders Kofod-Petersen)

2.2 Army creation

Each individual unit have a base cost associated with it, this cost represents how good the unit is. As the player adds better equipment, e.g.: weapons and armor to the unit, the cost increases.

When preparing for a battle the two players determine the total army points for each player to be used during the game. Common values are 1000, approximately an hours worth of game play; 2000-3000 points, for an evenings battle; and 4000 or more points, for an entire day (Games Workshop, 2009, p.132). To create the army the player selects the units he/she wants to use for the battle while using as many points possible, but still under the total allotted points. As it is difficult to use exactly all the points, both armies are usually of a different value, but as close to the limit as possible.

There are however a few rules that must be followed when creating the army, you cannot choose ten of the meanest unit there is and be done with it. The player should first select a *General*, which represents the player and leads the army on the battlefield. A General must be either a Lord or Hero unit. The rest of the army is selected based on rules governing the following unit categories: *Lords*, *Heroes*, *Core Units*, *Special Units* and *Rare Units*. All equipments and/or mounts purchased to individual units or groups, counts towards the total point usage within each category. If each unit in the group is meant to get the purchased item is the cost cumulative; E.g.: Shields with a cost of 1, purchased for a group of 20 Spearmen, will add a cost of 20 to the group. There is also restrictions on how many units of the same type can be included in the army (Games Workshop, 2009, p.134-135).

As *Lords & Heroes* are described in Section 2.1 it is only mentioned that you can spend up to 25% of the army points on Lords and Heroes respectively, the *no limit* under duplicate choices are not entirely correct, most lords and some heroes are unique and can only be included once in each army. *Core units* are the basic units of your army and the ones you will have the most of, e.g.: basic footmen and cavalry. You must use a minimum of 25% of the army points on core units. *Special units* are elite troops which perform great on their own or, as motivators for lesser soliders. You can spend as much as 50% of the points on these units. *Rare units* are very powerful units, e.g.: monsters, weird war machines and elite soliders of unsurpassed skill. You can use up to 25% of the points on these units. When choosing special and rare units there is a limit to how many *duplicates* you can have of these units. Special units can have three units of the same type while rare can only have two, i.e.: you can only have two dragons, but you may also have other rare units. The final rule governing the army creation reflects how many formations the army must consist of: An army must have at least three unit formations in addition to any lords and heroes in the army. “An army just isn’t a army unless it has plenty of warriors in its ranks” (Games Workshop, 2009, p.134). These rules are summarized in Table 2.2.

Table 2.2: Army creation summary (Games Workshop, 2009, p.135).

| | | |
|--|---------------------|---|
| The army must consist of at least three units, and one lord or hero to be the general. | | |
| | Points limit | Duplicate choices |
| Lords | Up to 25% | No limit |
| Heroes | Up to 25% | No limit |
| Core | 25% or more | No limit |
| Special | Up to 50% | Up to 3 (6 if you use 3000 or more army points) |
| Rare | Up to 25% | Up to 2 (4 if you use 3000 or more army points) |

Chapter 3

Design

This chapter describes aspects of the design and reasoning behind that design as well as a description of the technologies used. First is the evolution of the domain model explained, where every major iteration of the design process is described in Section 3.1. Then in Section 3.2 is the fundamental design described, before the chapter is concluded in Section 3.3 with a description of the technologies used by the system.

3.1 Domain model

The data in the domain model are a collaboration of two data sources: official warhammer rulebooks (Core rulebook and specific army rulebooks) and data from real-world battles fought and recorded by WarTrond (Trondheim Warhammer club). Data from the official rulebooks are all the general domain knowledge acquired, this knowledge includes: playable races (e.g.: Dwarfs, High Elves, Skaven and Lizardmen); race specific units, equipment and mounts; general equipment; and the rules necessary to create an army. The data from WarTrond are used to model the initial cases into the case base and this data consists of: The army composition of the player (i.e.: what race, units and equipment were used by the player); the number of points available to create the army; the opponents race and the outcome of the match.

The domain model have been through several development phases and each phase is described along with the reasoning behind the model in Sections 3.1.1 to 3.1.3.

3.1.1 Phase 1

In this phase were the ultimate goal to create a small straightforward model with only basic information and a simple similarity measure. This domain model consisted only of the cases (Player race, opponent race, outcome, player units) strictly encoded into it with no focus on normalization or reuse of data. When adding a few (two) cases to the case base and querying with advantageous data (i.e.: create queries that fit the small domain) I found that the calculated similarity coincided adequately with the target

cases. Since this were a simple calculation, where much of the information was ignored, I found that the most important feature (in this simple similarity calculation) are the number of available army points. As a result would the most similar case be classified by the points even though the case had the wrong opponent and even resulted in defeat.

By implementing and testing this first phase several lessons were learned; opponent race and outcome must be weighted high and most probable exclude the case if it is wrong. The amount of points, while important should not be responsible for the majority of the similarity value, and then have less weight than in the first phase. The last lesson (and one I were aware of from the beginning) is that this simple scheme and model have far to little information to be of any help during the explanation phase.

A description of the database schema depicted in Figure 3.1 follows. Case is the primary table containing the case ID, player- and opponent race, outcome and army points. The Case_Unit-, Unit_Equipment- and Unit_UTILITY tables are many-to-many relational tables. The one and two letter fields in Unit and UtilityUnit denotes Warhammer unit information that were encoded into the case structure.

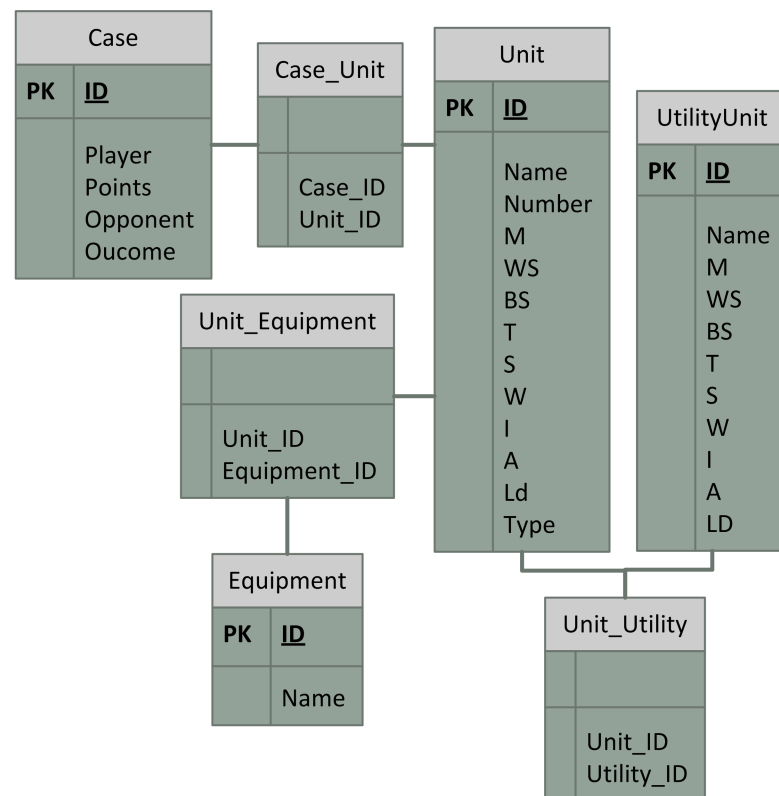


Figure 3.1: Phase 1 database schema

3.1.2 Phase 2

After the “successful” test in phase 1 a new database model were created; where focus were put into creating a more detailed and more normalized database; which could be used to properly describe a case and supply more explanatory power to the system. Figure 3.2 depicts the new and improved database schema. When this new schema was completed, the implementation were postponed until a discussion had been held to ratify it. The results of the discussion is transcribed below.

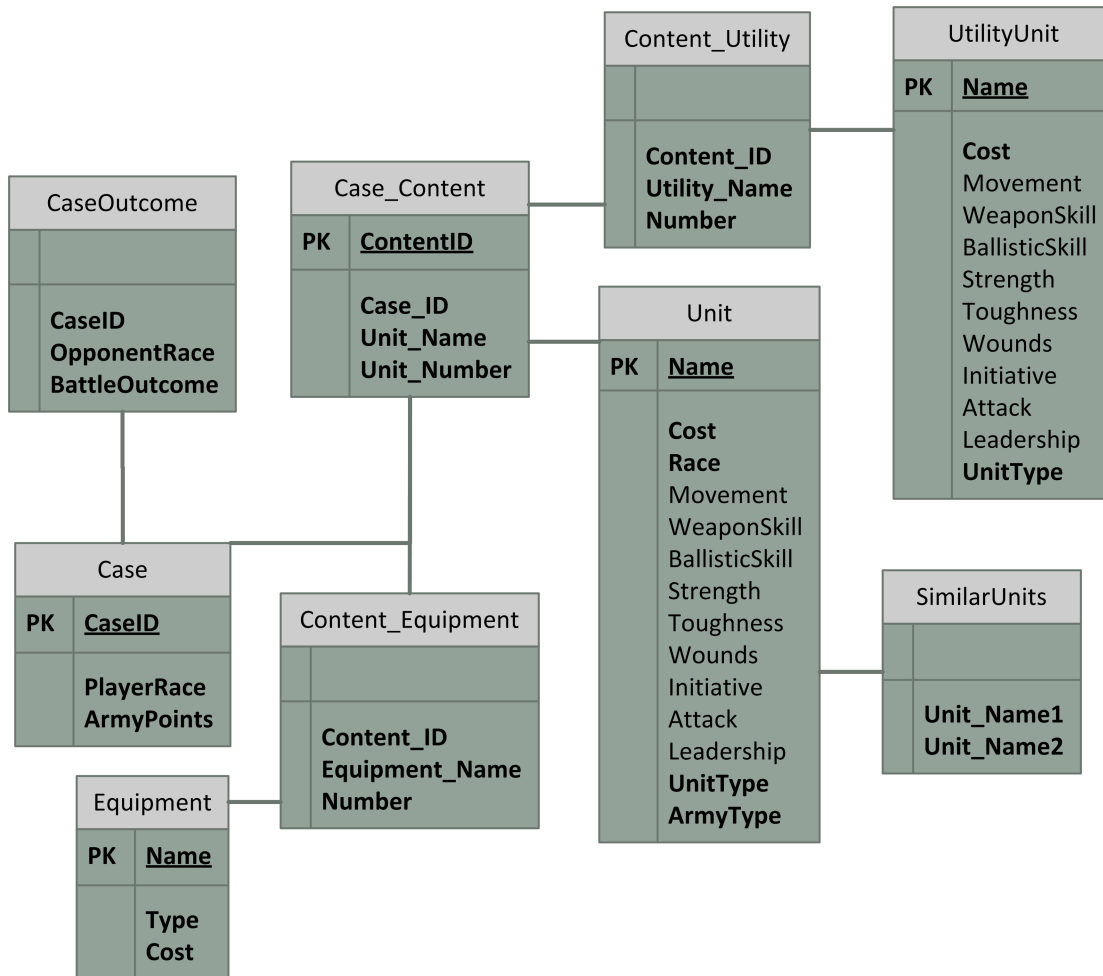


Figure 3.2: Phase 2 database schema

Decided to drop the SimilarUnits table, this table were intended to help the system know that Unit(X) is similar to Unit(Y) when it comes to unit stats and usage area on the battlefield. The key reason this table were dropped is because such static information could result in poorer similarities and army lineups. By keeping this information

dynamic and incorporating it into the reuse step of the CBR-cycle we would get the same functionality as the table and we can use this dynamic information to improve the justification parts of an explanation e.g.: 20xHelbadiers is similar to 15xSpearmen with a similarity of 78%, but 1xSteam Tank have a similarity of 0.01% with 15xSpearmen¹.

Decided to connect the equipment and utility units to the unit table through many-to-many relational tables instead of the existing connection to Case_Content, this is because each unit should be fully aware of what equipment and support units it possesses.

Decided against the removal of unit base stats, since these stats can be used in the reuse and explanation phases in order to adapt and justify the changes made to the initial case. This decision is linked to the decision to drop the SimilarUnits table.

Decided to add special rules associated with units into the domain model, this information is important when a battle is fought and as a result to some extent useful when the army lineup is made, but the usage of this information in the system is reserved for future work on the system, or if time suffice.

This schema were never implemented in full, as apparent shortcomings or redundancies were discovered while working on the implementation; these shortcomings consisted of improperly described equipment, missing database relations and missing normalization. The changes introduced by these discoveries and the above discussion resulted in the database schema presented in the next section which is the schema used in the system implementation.

3.1.3 Phase 3

This section describes the database schema depicted in Figure 3.3. Some aspects of this schema is similar if not identical to the schema discussed in Section 3.1.2; the changes being the result of the discoveries made during the discussion and partial implementation of schema 2.

Additional fields were introduced in the the underspecified Equipment table to more accurately convey the necessary knowledge about the equipment: Range, Modifier, UsableBy and DefaultEq. It could be argued that the *Range* and *Modifier* fields should be merged together as they both is used to describe the properties of the item; but since range is a specific concept that relates to how far a ranged weapon (e.g.: bow or pistol) can be fired, or the AoE of an enchanted item/battle standard; and the modifier is used to convey if the item have a special effect, enhance the units core attributes (e.g.: strength or attack) it was decided to keep them apart. *UsableBy* is used to denote if all races can use the item or to restrict it to a special race as most items are unique to a race and *DefaultEq* denotes if the item is belonging to a unit by default.

The Army_Unit_Equipment and Unit_Equipment tables convey parts of the same knowledge, both are related to a unit and that units equipment. The differences and reason for having both is that the Unit_Equipment table dictates what equipment a unit *can* equip while the Army_Unit_Equipment table specifies what the unit *have* equipped

¹These similarites are not accurate and only provided to demonstrate the example

in this army configuration. Similar tables and reasoning exists for the different utility units.

In the Unit table are the following fields added: MinNumber, MaxNumber, MagicPoints & WeaponType, with this information is the unit fully in control of all the knowledge associated with that one unit. The *min*- and *max* numbers indicate the minimum and maximum number of units a group (formation) must/can have in order to comply with the game rules, a max number of 0 indicates that there is no legal upper limit to the formation size. Some units have the possibility to buy additional magical items to use and the *MagicPoints* field indicates how many point that unit can spend on those items. Finally is the *WeaponType* field used to determine the primary weapon type of the unit (excluding purchases of magical items), the available values are: *melee*, *ranged*, *long weapon* and *great weapon*. A unit is characterized as *melee* unless one of the other are true. This information is used to calculate the similarity between units.

In UnitUtility table were the Required, NumUnits & PromotionUnit fields added. *Required* is used to inform the program that this utility unit is required to be attached to the main unit (e.g.: a crew is often attached to a war machine). *NumUnits* indicate how many units the utility unit consist of (e.g.: a war machine crew typically consists of three units), and *PromotionUnit* dictates that the utility unit is an upgraded version of the main unit (e.g.: a crossbowman may be upgraded to a marksman), upgraded units typically have one of their characteristics incremented by one.

Finally is Armies and Army_Units new tables, Armies is used to hold the player race and army point values, the armies id is in turn used to bind units to that particular army and to bind the army to one or more cases. The Army_Unit table have all the information about the unit setup in a particular army, which units are present, what equipment they have, which utility units they utilize and how many unit there are in each formation.

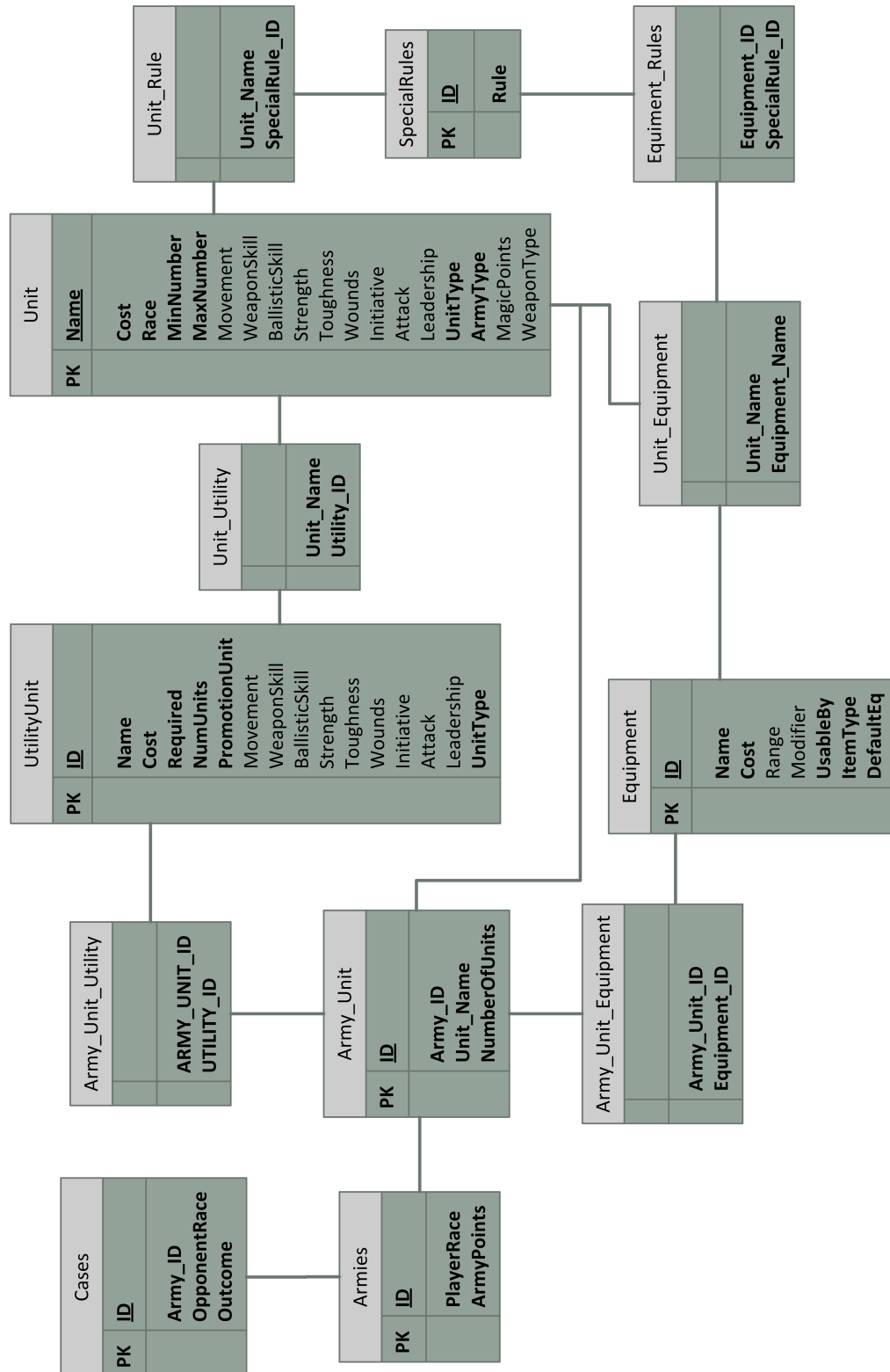


Figure 3.3: Phase 3 database schema

3.2 Fundamental design

This system is called *Myrmidia*; she is the patron goddess of warfare, war craft and soldiers in some areas of the Warhammer Fantasy world, which makes her the perfect deity in which to dedicate this system. The system consists primarily of four components: Its knowledge (*domain model*), the *CBR* component, the *explanation* component and the *Warhammer* component. The domain model were thoroughly described in the previous section (Section 3.1). The other components will be covered in greater detail in the implementation chapter (Chapter 4) with the exception of the CBR component which is partially covered in Section 3.2.1, by revealing the case structure and the versatility of the queries.

3.2.1 Case structure

jColibri supports the distinction between the problem description, solution and justification of the case. All three distinctions are included in the CBRCase class supplied by the framework. In Myrmidia however are there no distinction between the problem description and its solution, as both are contained within the same class structure and essentially contains exactly the same features.

A query must at the very least contain features indicating which race the player wants to use, the opponents chosen race and the army points to be used to create the army, as indicated in Table 3.1.

Table 3.1: Example of a simple case query

| | | |
|------|----------------|--------|
| Case | Player race: | Empire |
| | Opponent race: | Skaven |
| | Army points: | 2500 |

While the simple query contains all the information the system need to create a query and retrieve the similar cases from the case base, it does cast a wide net. By taking the time to specify more information the query grows more restrictive and the results from the similarity calculations might differ. In Table 3.2 is an example of how the query have grown to accommodate other features. There is no restrictions on how many Case-Units the query can contain, nor what legally equipable equipment is supplied in each unit; Figure 3.4 displays the user interface to create these queries. There is no guarantee that the requested units and/or equipment/mounts is present in the final solution, as the automated adaptation have few constraints when it tries to change the case to fit both the query and the game rules; where the game rules have precedence over the query.

Table 3.2: Example of a more extensive case query

| | |
|-----------|---|
| Case | Player race: Empire Opponent race: Skaven Army points: 2500 |
| Case-Unit | Name: Crossbowmen Formation size: 10 |
| Case-Unit | Name: Arch Lector Formation size: 1 Equipment: Mace of Helstum Armour of Fortune Potion of Speed Utility unit: Barded warhorse |

There are a few factors that led to the decision of not separating features and creating description/solution features where the most prominent of these are:

1: Difficult to categorize the features and by so doing restrict them to be a description or solution feature.

2: It felt restrictive and impairing on the functionality of Myrmidia and the users possibility to define the *perfect* query for their needs.

3: All the features feels like they belong in both the query and the solution, by being unable to separate the features two options remained: (1) to create two case hierarchies where all features are present in both, or (2) to have the query (problem description) and solution be part of the same hierarchy. The latter were chosen in an effort to minimize the code; and to create two identical case structures where the only difference are an implied conceptual difference, is unnecessary.

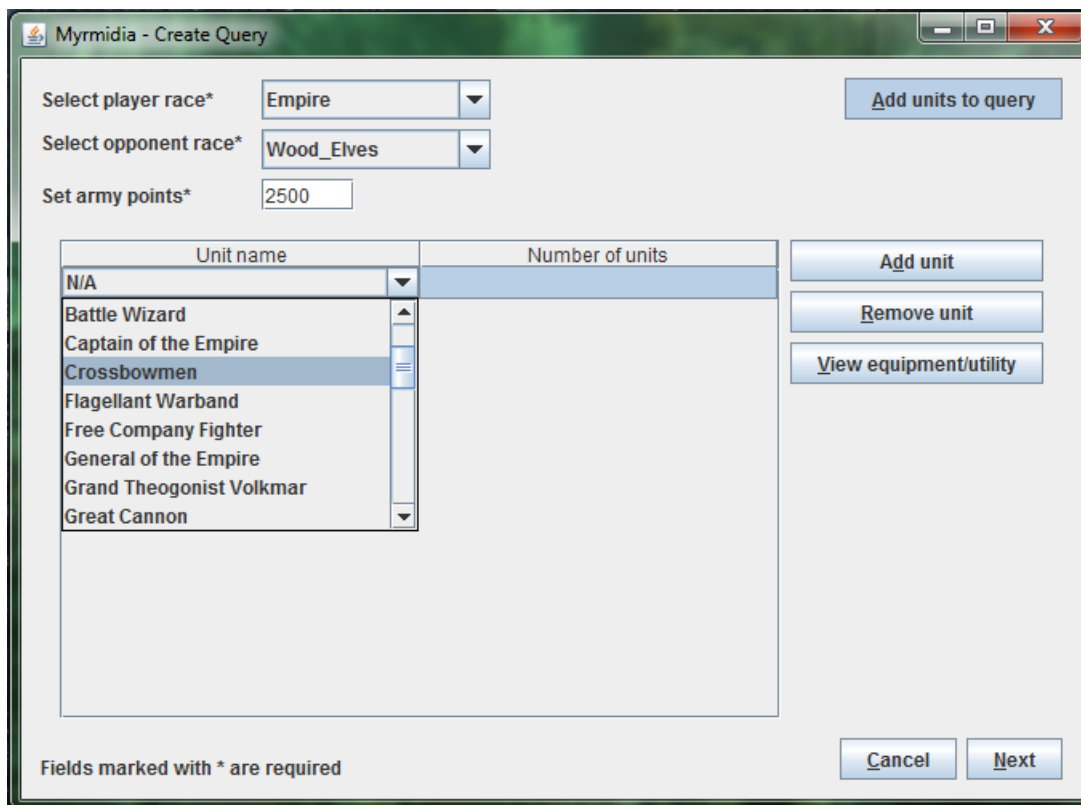


Figure 3.4: The create query UI

3.3 Technologies used

This section describes the various technologies utilized in Myrmidia and the reason for using them. jColibri is described in Section 3.3.1, while Apache Derby is described in Section 3.3.2 and finally is Hibernate described in Section 3.3.3.

3.3.1 jColibri

jColibri is a CBR framework for Java developed and maintained by the Department of Software Engineering and Artificial Intelligence at Complutense University of Madrid. Myrmidia uses jColibri version 2.0, but a third major release is forthcoming². The framework is fully implemented in Java and includes several out-of-the-box features and extensive interfaces to create new functionality and/or tweak the functionality to fit the project under development. jColibri is easy to setup and integrate into a project since all configuration is done through a few XML files. As it supports a wide range of database

²April 2011 - jCOLIBRI 3 & COLIBRI Studio almost ready: <http://gaia.fdi.ucm.es/projects/jcolibri/> Accessed: 2 May, 2011

drivers and utilizes Hibernate for POJO mappings is the framework available for use by most Java CBR applications. Additionally can ontologies be included through the use of the OntoBridge functionality, which simplifies the use of ontologies to create knowledge intensive CBR applications (Garcia et al., 2008).

Through the use of interfaces in all core components of jColibri, can any application specific classes be easily and seamlessly integrated into and used in conjunction with features out-of-the-box. The similarity calculations are divided by what is called local- and global similarity functions (Díaz-Agudo and González-Calero, 2001). The *local* similarity functions are used to calculate the class specific similarity or even feature specific similarity (E.g.: Object *X* have two features *A* and *B*. A local similarity function is assigned to calculate the similarity of feature *A* and another for feature *B* while a third function calculates the similarity of Object *X* with the two features as variables in the computation). The *global* similarity function calculates the case similarity (E.g.: Objects *X*, *Y* and *Z* are objects of different classes which all contribute to the case similarity; and each have one or more different local similarity functions attributed them. The global similarity function calculates the case similarity by using the calculated local similarity of the three objects as variables in the computation).

jColibri in Myrmidia

jColibri were chosen because of its ease-of-use framework, and the effort could then be moved from *how to make it work* to *how should this be implemented?* In Myrmidia is most of the jColibri functionality changed or adapted from the initial framework operations in order to fit the projects needs. While some of the functionality is mere clones with one line of code added/removed, others are completely written from scratch (through the use of interfaces) to achieve the desired functionality. Most of these changes are with the local- and global similarity functions and these are more thoroughly described in Section 4.1.2.

3.3.2 Apache Derby

The Apache Derby database solution is a open source, full-featured relational database management system (RDBMS), based on Java technology (JDBC) and SQL. Apache Derby is written entirely in Java, this makes any software based on Apache Derby and Java, as platform independent as Java is by itself. Additionally is the database files generated by Apache Derby, platform independent. This means that the database folder (with data) can be copied onto any Java enabled platform and be accessible by the application without the need to change the data or data representation (Apache Software Foundation, 2010).

Two database driver modes are supplied: a server/client mode and a embedded mode. The *server/client* are the default configuration and is the standard database approach. The database is located on a server and all the clients (programs) access that server to acquire data from the database. *Embedded* mode does not require a server to run the database, nor a separate process on the client machine; the database management

system runs within the same JVM as the “client” (program) which requests the data. Figure 3.5 displays how the embedded Apache Derby architecture is bundled within the JVM. There is one major drawback by using this embedded approach, only one client on the host machine may have a connection with the database at any time, that one JVM may have many connections, but another JVM process will get an access violation and be prevented from accessing the database.

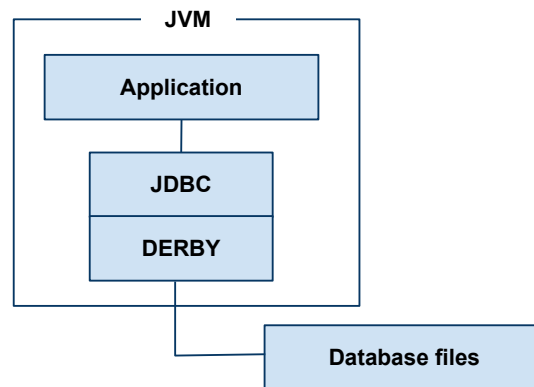


Figure 3.5: Embedded Derby Architecture http://db.apache.org/derby/papers/DerbyTut/embedded_intro.html

Apache Derby in Myrmedia

Apache Derby with the embedded mode activated is the current RDBMS in Myrmedia. The ease of use, bundled with no database server requirement nor a separate RDBMS process on the host machine, made it ideal for prototyping the database and Myrmedia in general. By bundling the database folder together with the application will any user be able to use a local copy of Myrmedia without the need to setup his own database server or being connected to the Internet.

A known drawback which resolution is reserved for future work is that: Since each user have his/hers own local copy of the database, will the CBR portion of Myrmedia be restricted to learn from that one user, instead of the whole user base. This limits the learning capabilities of the system and prevents more uniform results across the user base.

3.3.3 Hibernate

Hibernate is a Object/Relational mapping tool developed in Java and designed to simplify the process of mapping data in Java objects to database tables and vice versa. Mapping information encoded in XML files or as annotations in the Java class files provides Hibernate with all the necessary information to map data between database and POJO objects. To enable Hibernate mapping for a class must all class variables, which

is to be mapped, have getter and setter functions. A huge advantage with Hibernate is that although the persistence process is automated and hides much of the work for the user, the underlying technology is not hidden. Developers can execute custom commands through either the Hibernate Query Language (HQL) or through native SQL (Bernard et al., 2011).

The below frame illustrates a simple example of Hibernate XML mapping for the class depicted in Figure 3.6. Each property in the Person class is mapped with its name and the name of the column in the Person table to map the data.

```
<hibernate-mapping>
  <class name="Person" table="Person" >
    <id name="id" type="integer" >
      <generator class="increment" />
    </id>
    <property name="firstName" column="First" type="varchar(30)" />
    <property name="lastName" column="Last" type="varchar(30)" />
  </class>
</hibernate-mapping>
```


|  Person |
|---|
| <i>Attributes</i> |
| private int id private String firstName private String lastName |
| <i>Operations</i> |
| public Person() public int getId() public void setId(int val) public String getFirstName() public void setFirstName(String val) public String getLastName() public void setLastName(String val) |

Figure 3.6: An example class for Hibernate mapping

Hibernate in Myrmidia

Since Hibernate is the default persistence tool in jColibri, the jColibri tutorial (Garcia et al., 2008) uses Hibernate and finally since Hibernate is a powerful persistence tool it is chosen in Myrmidia. In Myrmidia is hibernate used for the initial data fetching when a CBR query is posted to the system, and when HQL queries are sent to fetch additional data during CBR query construction and result adaptation. Because of difficulties with

configuring Hibernate properly it is however not responsible for the storing of new cases into the case base. For that task the query is first passed through a preparation stage which updates all the case data IDs to be new unique IDs and then parsed into native SQL. This is not ideal, but Hibernate insisted in trying to update existing cases instead of creating new ones, in addition to insert data into “static” tables. After several attempts to correct these errors it was decided to remove that part of the persistence functionality and instead perform that mapping manually.

Chapter 4

Implementation

In this chapter will most of the implementation work be described through examples, class-diagrams and equations. Not every aspect of the application will be described in this way, but rather the major components and how they interact within those components. The components/packages that will be described is CBR in Section 4.1, explanations in Section 4.2 and some of how Warhammer and a case is represented in Section 4.3. The sections for CBR and explanations will begin with a brief introduction to those concepts as they not are discussed elsewhere in this thesis.

4.1 CBR

The inspiration for CBR came from a desire to understand how people remember information and are in turn reminded of information; and that subsequently it was recognized that people commonly solve problems by remembering how they solved similar problems in the past.

-Watson (1998)

The standard definition of CBR was formulated by Riesbeck and Schank (1989): “A *case-based reasoner solves problems by using or adapting solutions to old problems.*”

The CBR component of this application consists of the four steps in the CBR cycle as described by Aamodt and Plaza (1994). They stipulate the four steps as *Retrieve*, *Reuse*, *Revise* and *Retain*; where *Retrieve* is the process of acquiring the cases from the case-base; *Reuse* is the process of using and adapting the acquired data to fit the new problem; *Revise* is the process of evaluating the solution, testing it and finally determine its usefulness; and finally *Retain* is the process of storing the relevant pieces of data back into the case-base. Figure 4.1 depicts the CBR cycle as described by Aamodt and Plaza.

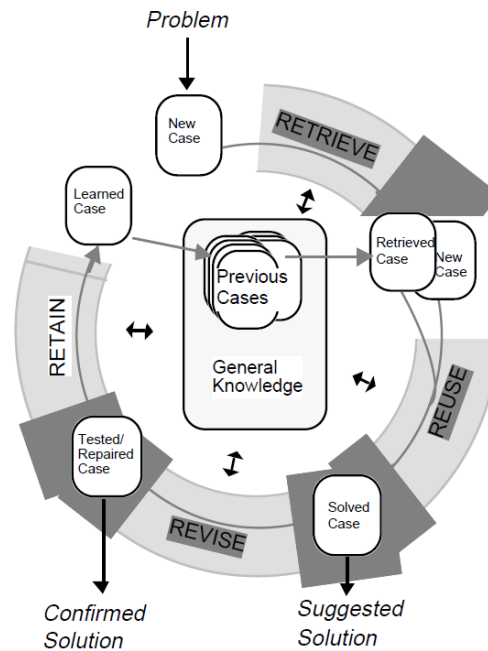


Figure 4.1: The CBR cycle (Aamodt and Plaza, 1994)

The remaining parts of this section describes how Myrmidia have implemented CBR, beginning with the similarity functions utilized in the retrieve step to limit the cases into a manageable sub-set of cases based on the *k-NearestNeighbour* (kNN) algorithm in Section 4.1.2. Section 4.1.3 describes the similarity function and adaptation rules used during the reuse step, while the revise and retain steps are described in Section 4.1.4.

4.1.1 CBR framework

Most of the CBR framework is presented in rich detail in the subsequent sections (4.1.2, 4.1.3 and 4.1.4 and is displayed in Figure 4.2; In order to preserve readability of the class diagram are all attributes and operations hidden. The components in the figure are not explicitly explained, but rather implicitly linked to the explanation of the CBR-, similarity- and adaption processes.

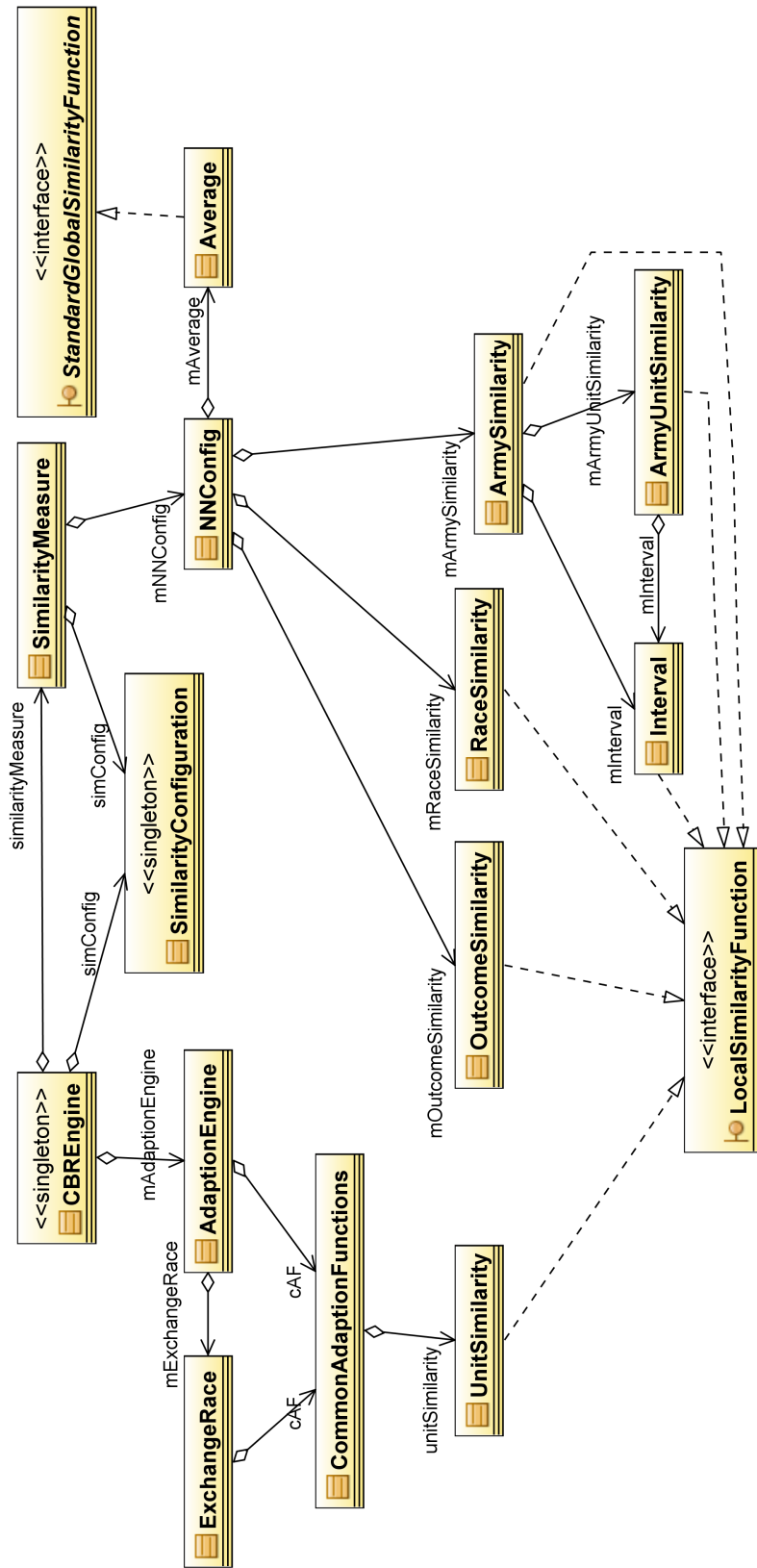


Figure 4.2: CBR class diagram

4.1.2 Similarity (Retrieve)

First off is all valid cases in the case-base collected, then the similarity calculations matches the retrieved cases with the query posted to the system, in an effort to find the *k nearest neighbouring* cases. The kNN algorithm were chosen since it quickly can distinguish between cases based on the similarity between them.

The similarity is calculated based on a weighted sum of three components: army, opponent and outcome. Of these three are the army component the most complex and is in itself a result of several weighted similarities. The different equations used to calculate the similarity are presented below. All the equations presented here will use abbreviations on the variable names to limit the equation space requirements. All abbreviations will follow the same naming conventions to be easily understandable, and each abbreviation will be explained in the associated equation text, an *W* following the abbreviation will always mean that the variable denotes the variables weight in the calculation.

Equation 4.1 is the equation used to calculate the weighted average of the three components. A = “Army”, Opp = “Opponent” (the opponent race), and Out = “Outcome”; which constitutes the entire case.

$$\text{Similarity} = \frac{\sum A \times AW + \text{Opp} \times \text{Opp}W + \text{Out} \times \text{Out}W}{\sum AW + \text{Opp}W + \text{Out}W} \quad (4.1)$$

Equation 4.2 calculates the first component in Equation 4.1 and is also a weighted average of its components. This equation calculates how similar the army in the case is to the army in the query. A = “Army”, PR = “Player race” (the similarity between the race the player wish to use and the race in the case), AP = “Army points” (similarity between the desired army points and the case army points) and AU = “Army unit” (the similarity between units in the query and units in the case).

$$A = \frac{\sum PR \times PRW + AP \times APW + AU \times AUW}{\sum PRW + APW + AUW} \quad (4.2)$$

Equation 4.3 is used in both Equations 4.1 and 4.2 to calculate the similarity of the query race and case race as either the opponent or the player. Race equals *Opp* (in Equation 4.1) or *PR* (in Equation 4.2) depending on the context in which the equation is used.

$$\text{Race} = \begin{cases} 1 & \text{if query race equals case race} \\ 0 & \text{if query race not equals case race} \end{cases} \quad (4.3)$$

Equation 4.4 is used in Equation 4.1 to calculate the similarity between the desired outcome (victory) to the actual outcome of the case.

$$\text{Out} = \begin{cases} 1 & \text{If the query outcome equals the case outcome} \\ 0.5 & \text{If the query = Victory and case = Draw, and vice versa} \\ 0 & \text{If the case equals defeat} \end{cases} \quad (4.4)$$

Equation 4.5 is used in Equation 4.2 to calculate the similarity between the case- and query army points. The case or query may contain the wild card value 0, which means that the case does not know how many points were used to create it, or the player does not know how many points he should use. The former are a method to help cope with poorly defined cases (cases missing attributes) and the latter is a *what if* scenario, which probably never will occur. The similarity is based on the difference between the case- and query points, divided by the range(interval) in which the similarity should be valid. By default is the interval set to 500 points.

$$AP = \begin{cases} 1 & \text{If the query- or the case army points equals 0} \\ x \in [0, 1] & 1 - \left| \frac{\text{Case points} - \text{Query points}}{\text{Interval}} \right| \end{cases} \quad (4.5)$$

Equation 4.6 is used by Equation 4.2 to calculate the similarity between the units in the query and units in the case. AU = “Army unit”, UF = “Unit fraction”, NF = “Number fraction”, EF = “Equipment fraction” and UtF = “Utility unit fraction”. AU is the fraction of unit in the query found in the case e.g.: There are four units in the query and two of them are present in the case will give a unit fraction of: $\frac{2}{4} = 0.5$. The denominator is not a fixed value as is the case with most average calculations, but is set to reflect the number of unknowns in the equation which is greater than 0. This is made deliberately to prevent the similarity from being punished if the case contains equipment and/or utility units and the query does not. E.g.: If the query does not contain any items and utility units reduce the *denominator* from 4 to 2.

$$AU = \begin{cases} 1 & \text{query does not specify any units} \\ 0 & \text{none of the query units are present in the case} \\ x \in [0, 1] & \frac{UF+NF+EF+UtF}{\text{Denominator}} \end{cases} \quad (4.6)$$

Equation 4.7 is used by Equation 4.6 to calculate the fraction of the sum of the queried equipment, found in the case e.g.: The query contains three units with 2 items each, the case contains all three units, but with only one of the queried items each will give a equipment fraction of: $\frac{3}{6} = 0.5$. The exact same formula is used if/when a query contains units with utility units.

$$EF = \frac{\sum_{i=1}^n \text{found equipment in unit}_{[i]}}{\sum_{j=1}^n \text{queried equipment in unit}_{[j]}} \quad (4.7)$$

Equation 4.8 is used by Equation 4.6 to calculate the fraction of the sum of the queried unit formation sizes. The similarity is based on the difference between the case unit formation size and the query unit formation size, divided by the range(interval) the similarity should be valid. By default is the interval set to 5. To illustrate the

equation imagine a query which contains a formation of 20 archers and a formation of 20 swordsmen. The case have a formation of 20 archers, the swordsmen formation contains only 15 units. The similarity for the archers would be $1 - \left| \frac{20-20}{5} \right| = 1$; for the swordsmen would the similarity be $1 - \left| \frac{15-20}{5} \right| = 0$; and the total similarity would be $\frac{\text{archer similarity} + \text{swordsmen similarity}}{\text{Found units}} = \frac{1+0}{2} = 0.5$.

$$NF = \frac{\sum \left(1 - \left| \frac{\text{Case unit formation size} - \text{Query unit formation size}}{\text{Interval}} \right| \right)}{\text{Number of query units found in case}} \quad (4.8)$$

The calculated similarity is then used by the kNN algorithm to sort the cases in descending order and to collect a sub-set of the k cases with the highest similarity. All the weights used by the similarity calculations above can be configured by the user before the query is sent to the retrieve step along with the variable k .

All the weights in the above equations can be set by the user through the user interface depicted in Figure 4.3, as well as the number of cases to retrieve (the k in the kNN algorithm) and the interval in which army points similarities are valid.

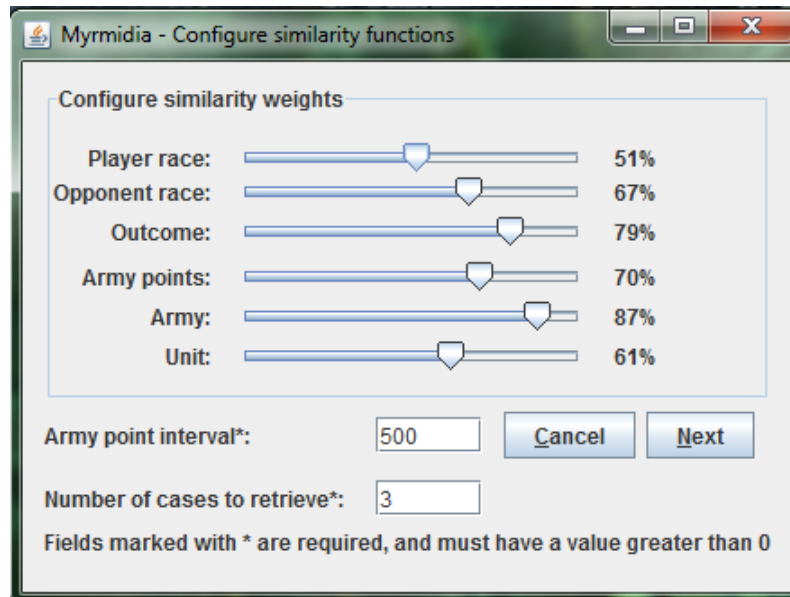


Figure 4.3: Similarity weight configuration UI with random weights

Retrieval results

In order to give the user more control over the process and the results of the case retrieval, the results are displayed in its own user interface prior to adaption. This gives the user the possibility to accept or decline cases which in his/hers eyes are inappropriate, but no

manual changes are permitted at this time, since the software is dealing in partially static data. Figure 4.4 displays the user interface and the retrieval result of the query. To help improve the users confidence and to help approve the cases is this where transparency (more details in Section 4.2.1) explanations can be viewed.

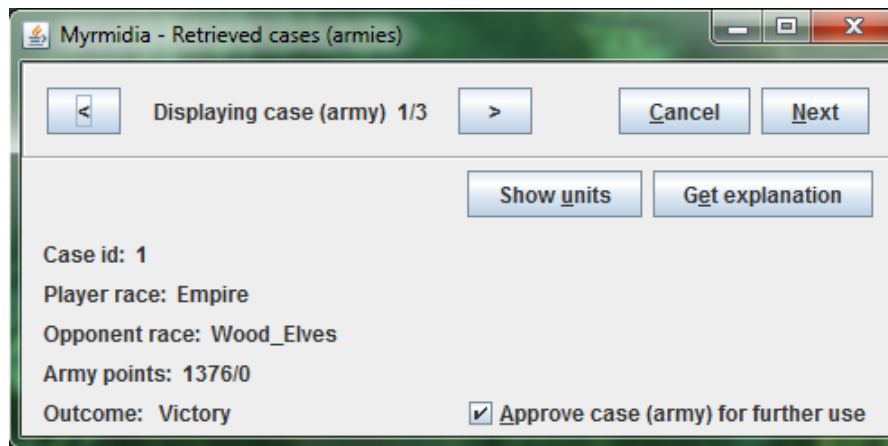


Figure 4.4: The retrieval result UI

4.1.3 Adaptation (Reuse)

The reuse step of the CBR-cycle is initiated when one or more of the retrieved cases (from the kNN algorithm) is approved by the user. A retrieved case is first passed through a *naive* adaption before a more *exhaustive* adaption *may* be performed.

The naive adaption is used to exchange data in the query with data in the case, the data exchanged is effectively the player- and opponent races, the army points and any units in the query is exchanged based on the following logic:

```

for each unit in query do
  if unit exist in case then
    case unit size = query unit size
    case unit equipment = query unit equipment
    case unit utility unit = query unit utility unit
  else
    find most similar unit
    case most similar unit = query unit
  end if
end for

```

Finally is the outcome of the case set to unknown.

When the compulsory naive adaption is completed the exhaustive adaption process is started, this process may or may not perform any adaption on the retrieved cases. The amount of adaption performed is dependent upon a set of rules and whether or not any of those rules are violated. The rules are based upon the general rules a person

must adhere to when manually creating a army roster and some rules created to help the CBR reasoner. Each rule in turn have a set of associated rules or actions performed if violated. Below is a description of each stipulated rule which may be violated and a brief description of the solution:

No army general: The army have no lord or hero units, or the only hero unit in the army is a battle standard bearer. As the army must have a general the solution to this rule is to add a new random hero or lord unit.

OK: No violated rules were found and the adaptation is completed.

Too many duplicate rare units: The rules stipulate that there is a limit on how many rare units there may be in the army. This rule is violated when the number of duplicates exceeds that limit and the solution is to delete any offending units from the case.

Too many duplicate special units: The rules stipulate that there is a limit on how many special units there may be in the army. This rule is violated when the number of duplicates exceeds that limit and the solution is to delete any offending units from the case.

Too few core points: The rule stipulates that the total amount of core points used is less than the required 25% of the available army points. There is encoded in a complex solution for this problem and several actions may be performed. The first course of action is to fire the *Too few groups* and *too few units in group* rule actions, since this rule is dependent upon that those two not are violated. If these actions do not solve the problem are there two other actions that may be performed. (1) Increase core group costs by adding units or giving the group full command. (2) Create a new core unit group.

Too few groups: The rules stipulate that there must be at least 3 unit groups in addition to any lords and heroes in the army, for it to be an army. The solution is to randomly select if a core, rare or special group should be added, and then find the unit within that group which is the least similar to existing units of that type in the case.

Too few points total: If the difference between used army points and available army points is greater than a threshold value of 10 points will this rule be violated. Depending on the difference may one of three actions be performed: (1) add units to existing groups, (2) add equipment/utility units to existing groups or (3) add a new random group (core, special, rare, hero or lord).

Too few units in group: There exists one or more groups in the army which violates its minimum group size requirement. Each unit in the Warhammer universe as they are described in the various Warhammer race rulebooks have a minimum number of units which is required in a unit group. The solution is to increase the

number of units in the group so that they is greater than or equal to the minimum group size.

Too many hero points: A maximum of 25% of the available army points may be used on hero units. If the number of hero points exceeds this limit one of the following actions are taken: (1) If the difference between used points and legal points is less than 50 points, find the most expensive hero unit and remove the equipment or utility unit, which reduces the difference closest to zero. (2) If the difference is greater than 50 points, remove the least expensive hero unit.

Too many lord points: A maximum of 25% of the available army points may be used on lord units. If the number of lord points exceeds this limit one of the following actions are taken: (1) If the difference between used points and legal points is less than 50 points, find the most expensive lord unit and remove the equipment or utility unit, which reduces the difference closest to zero. (2) If the difference is greater than 50 points, remove the least expensive hero unit.

Too many points total: If the total used points exceeds the available army points triggers this rule violation. There are several actions that are attempted in order to solve this problem: (1) Check if any of the other rules that govern the amount of points used are violated (too many hero, lord, rare or special points; too many units in group, too many duplicate special or rare units). If this don't solve the problem then depending on the difference between used points and available points can the following actions be performed: (2) Reduce the number of units in groups, (3) Remove equipment/utility units from a group or (4) Remove a random group.

Too many rare points: A maximum of 25% may be used on rare units, if this limit is exceeded the following action is performed: (1) first check if the *too many duplicate rare units* rule is violated and then perform the actions associated with that rule violation. (2) Depending on the difference between used points and available points either reduce group size or delete a group.

Too many special points: A maximum of 50% may be used on special units, if this limit is exceeded the following action is performed: (1) first check if the *too many duplicate special units* rule is violated and then perform the actions associated with that rule violation. (2) Depending on the difference between used points and available points either reduce group size, remove equipment/utility units or delete a group.

Too many units in group: There exists one or more groups in the army which violates its maximum group size requirement. Some units in the Warhammer universe as they are described in the various Warhammer race rulebooks have a maximum number of units which is required in a unit group. Other units have no max limit, only the theoretical limit imposed by the available army points. The solution is to decrease the number of units in the group so that they is less than or equal to the maximum group size.

Wrong race: This rule is fired if the retrieved case have another player race than the query case, and is the first rule returned by the rule verification process. The solution is to go through the retrieved case and exchange every unit in the case with its most similar counterpart in the query race. By default is the existing unit size kept, unless it would cause rule violations in regard to minimum/maximum unit sizes.

It is important to note than only one of the above actions is performed and then again only one part-action (if several actions are available) in each loop iteration. This is to make sure that the rules are validated after each change, since one rule violation may be fixed by fixing another rule, since many of the rules are interdependent on each other.

Similarity in the adaptation phase

The adaptation phase have its own unique similarity function used when finding the least similar and the most similar units. The main unit similarity equation is displayed in Equation 4.9 and as the case similarity in Equation 4.1 is it also a weighted sum of its components. An explanation of the abbreviated component names follow, an abbreviation ending i a *W* always denotes that components weight: US=Unit similarity, Ch = Characteristics, UT=Unit type, AT=Army type, C=Cost, W=Weapon type and M=Magician. Each of the components of the equation is further described below.

$$US = \frac{Ch \times ChW + UT \times UTW + AT \times ATW + C \times CW + WT \times WTW + M + MW}{ChW + UTW + ATW + CW + WW + MW} \quad (4.9)$$

Since the characteristic values are stored as strings (to enable notation of special values like 2D6 in addition to regular numbers) are there two different methods to calculate one characteristic (e.g.: strength, toughness and initiative) and the one used is dependent upon it being a number or a string. For the string similarity is a simple equality equation 4.10 used while an interval equation 4.11 is used to calculate the numbers, and the interval in which the similarity is valid is 2. Finally is the entire unit characteristic computed as the average of the similarity of the nine individual characteristics as in Equation 4.12

$$Char = \begin{cases} 1 & \text{The characteristic value is an exact match} \\ 0 & \text{The characteristic value is not a match} \end{cases} \quad (4.10)$$

$$Char = 1 - \left| \frac{\text{Existing unit characteristic} - \text{Replace unit characteristic}}{\text{interval}} \right|, \in [0, 1] \quad (4.11)$$

$$Ch = \frac{\sum_{i=1}^n Char_{[i]}}{9} \quad (4.12)$$

The cost (C) similarity is calculated as an interval similarity, the same way as each unit characteristic and Equation 4.11 can be used to convey the function, although the text inside the equation would be slightly different.

The unit type (UT) similarity is found through the lookup table displayed in Table 4.1, where the similarity between all the different unit types in Warhammer. These numbers are by no means found through any scientific approach, but is rather found through a *this feels appropriate* approach. The *feels appropriate approach* is used in all the similarity lookup tables in this section.

Table 4.1: Unit type similarity lookup table

| | Ca | Ch | In | MB | MC | MI | Mo | Sw | Un | WB | WM |
|----|------|------|------|-----|------|------|-----|-----|------|------|------|
| Ca | 1.0 | 0.75 | - | - | 0.65 | - | - | - | - | - | - |
| Ch | 0.75 | 1.0 | - | - | 0.50 | - | - | - | - | - | - |
| In | - | - | 1.0 | - | - | 0.65 | - | - | - | - | - |
| MB | - | - | - | 1.0 | 0.5 | 0.5 | 0.8 | - | - | - | - |
| MC | 0.65 | 0.5 | - | 0.5 | 1.0 | 0.5 | 0.8 | - | - | - | - |
| MI | - | - | 0.65 | 0.5 | 0.5 | 1.0 | 0.8 | - | - | - | - |
| Mo | - | - | - | 0.8 | 0.8 | 0.8 | 1.0 | - | - | - | - |
| Sw | - | - | - | - | - | - | - | 1.0 | - | - | - |
| Un | - | - | - | - | - | - | - | - | 1.0 | 0.45 | 0.45 |
| WB | - | - | - | - | - | - | - | - | 0.45 | 1.0 | 0.35 |
| WM | - | - | - | - | - | - | - | - | 0.45 | 0.35 | 1.0 |

The army type (AT) similarity is found through the lookup table displayed in Table 4.2.

Table 4.2: Army type similarity lookup table

| | Core | Lord | Hero | Special | Rare |
|---------|------|------|------|---------|------|
| Core | 1.0 | - | - | - | - |
| Lord | - | 1.0 | 0.75 | - | - |
| Hero | - | 0.75 | 1.0 | - | - |
| Special | - | - | - | 1.0 | 0.5 |
| Rare | - | - | - | 0.5 | 1.0 |

The weapon type (WT) similarity is found through the lookup table displayed in Table 4.3.

Table 4.3: Weapon type similarity lookup table

| | Ranged | Melee | Great weapon | Long weapon |
|--------------|--------|-------|--------------|-------------|
| Ranged | 1.0 | - | - | - |
| Melee | - | 1.0 | 0.5 | 0.3 |
| Great weapon | - | 0.5 | 1.0 | 0.55 |
| Long weapon | - | 0.3 | 0.55 | 1.0 |

Magician (M) similarity is the last component of the unit similarity equation and this is a simple boolean comparison which returns a value of 1 if the two units have the same magician value or a value of 0 if they differ.

4.1.4 Revise and Retain

The revise and retain steps in Myrmidia is somewhat merged together in order to achieve the desired performance from the system. They work in conjunction over two steps through separate user interfaces, in order to store the new cases and update the outcome. The two interfaces are named Revise and Retain respectively, but both perform parts of the two operations.

Revise

The revise step in Myrmidia is a completely manual task where the user can change almost every aspect of the case, but some restrictions does apply. The restrictions can be categorized as either *limitations* or *invalidators*. The limitations are merely features the user interface cannot change, but if changed would not invalidate the result; these features are the opponent race and the battle outcome. Invalidators on the other hand would, if changed, void the validity of the retrieved case; these features are the player race and army points. What can be changed: is what units are included in the army, how many of each unit and their equipment/utility units.

The easiest approach to an semi-automated revise step is a simple re-ranking of the cases based on their new data and then let the user perform the remaining manual tweaks. At the other end of the scale is a fully automated revision system, or a more thorough discussion on these points see Section 6.1.4.

When a user is satisfied with one or more of the cases and wants to test those in real world battles he/she then approves the case and it is stored into the case-base, and the storage process is one of the results of the revise/retain merge. This preliminary storing of the case cannot be acquired during the retrieve step. This safeguard is to prevent the system to base a new solution on unverified knowledge. Since the data is stored into a “inaccessible” vault can the application and computer be safely shut down while the game is played.

Figure 4.5 displays the user interface window presented to the user during the revise step. The interface permits the user to change the above mentioned aspects of the case.

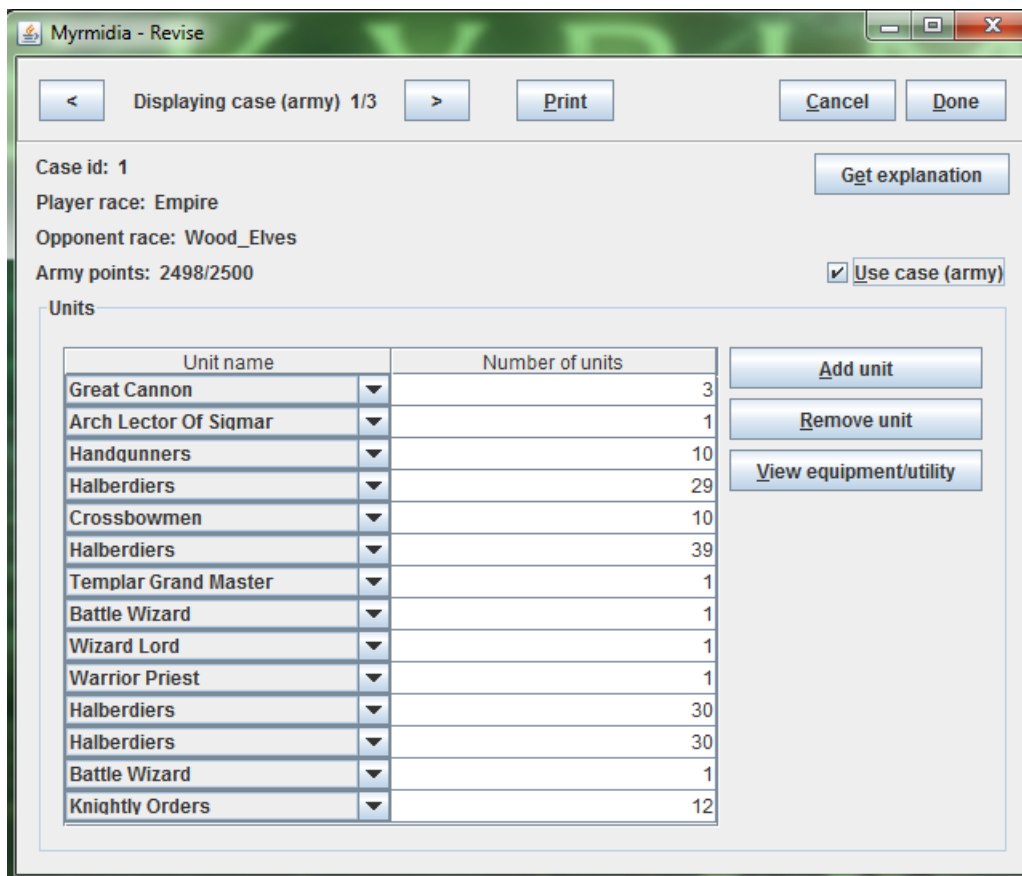


Figure 4.5: The revise UI

Retain

As a result of the partial merging between the revise and retain steps in the cycle is there one revision action that must be performed in the retain step; which is to set the result of the battle. Figure 4.6 displays the retain user interface and the outcomes a battle may have. The result determines what action the retain process performs in regard to the unknown case:

- If the case resulted in *victory* or a *draw* is that value updated in the case-base; which effectively removes the case from the case vault, and makes it available as a case in the system.
- If the case remains *unknown* it also remains in the case vault and remains inaccessible.
- Finally if the case resulted in *defeat* is that case deleted, what happens with the army represented in the case is however dependent upon one additional criteria. If the army in the case is used successfully in another case it will remain, if the

army however were exclusive to the defeated case will that army be purged from the database.

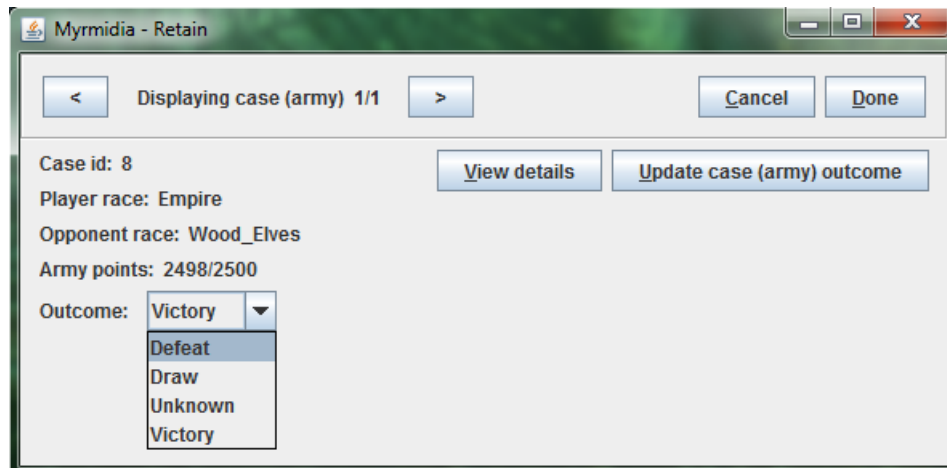


Figure 4.6: The retain UI

4.2 Explanations

Explanations are one of the primary human methods to reason and reach a solution to a problem. The purpose of an explanation is to explain a solution, the reasoning path that lead to this particular solution and how a system work, including how to handle the system. (Schank, 1986; Roth-Berghofer, 2004). Our use of explanations, its depth, scope and content are directly related to the users general domain knowledge, and the context in which the explanation is given. Explanations must be both *inclusive* and *instructive* (Roth-Berghofer, 2004). An explanation is divided into: *explanandum*, *explanans*, *facts* and *general laws*. Where explanandum is the question to be answered, explanans are the answer to the question, the facts and laws are what we know, and in turn base our explanation on.

Sørmo et al. (2005) describe five explanation goals for explanations in CBR applications: *transparency*, *justification*, *relevance*, *conceptualization* and *learning*. They do also point out that any explanation-aware CBR system are not bound to support all five goals, but rather a sub-set of goals dependent upon the systems need to explain. What knowledge the system have and/or what knowledge must be added to the system in order to fulfill a goal is also dependent upon the decision of which goals to support. *Transparency* is how did the system find the data it based its decision on. E.g.: In a CBR system, where cases are found based on their similarity values; how are those similarity values calculated and what are the similarity of the selected cases. *Justification* is to explain why the solution is a good solution, as a means to increase user confidence in the solution. *Relevance* is to explain how relevant features or questions asked by the system is towards creating the solution. *Conceptualization* the ability to explain concepts and

the terminology used by the system, and finally there is *learning*; Learning should be employed by systems designed to teach the user something about its domain, and while learning is represented to a certain degree in all the goals, is this goal only about learning. The learning should not be to teach the user how the system operates or reasons, as that knowledge may be abstract and difficult to obtain by a user, but rather how the user can perform similar functions in terms the user can understand.

Richter introduced the concept of knowledge containers (Richter, 1995), and Roth-Berghofer builds on this concept (Roth-Berghofer, 2004). Knowledge containers is a method to compartmentalize the knowledge contained in the CBR knowledge-base; there are four such containers: *Vocabulary*, *Similarity measure*, *Adaptation knowledge* and *Case-base*. Figure 4.7 depicts the interrelationship between these four containers. The vocabulary container is the all-encompassing container and contains all the defining data in the system e.g.: attributes, predicates and domain structure. While the three remaining containers are quite self-explanatory, are a brief description given; The similarity measures container know how to calculate the similarity between cases and query; The adaptation knowledge container, how to adapt stored cases to fit the description/problem; and the case-base container contains the actual cases the system consists of. Individually can these containers explain quite accurately their own role in the process, but more importantly their combined knowledge can satisfy most if not all of the goals outlined by Sørmo et al. (2005).

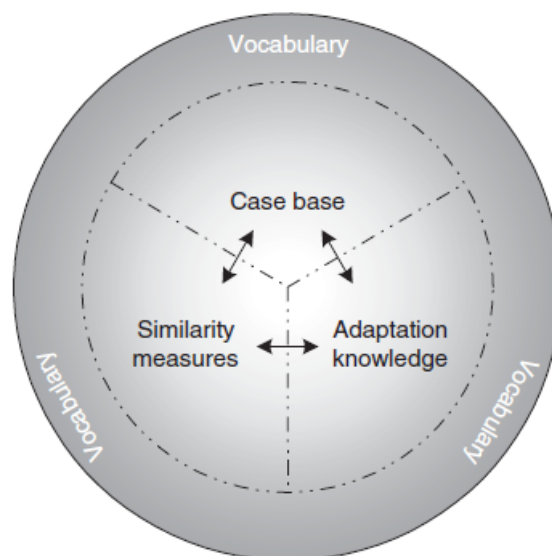


Figure 4.7: The CBR knowledge containers (Roth-Berghofer, 2004)

4.2.1 Explanation framework

An explanation engine were built, which in conjunction with several task-dedicated classes and an explanation interface, tracks all changes and operations performed on each retrieved case in order to generate the desired explanations. Figure 4.8 displays the class diagram for the explanation classes, to keep the diagram legible is all attributes and operations hidden.

The Explanation interface contains only one method *generateExplanation()* which returns the explanation string generated by the implementing component. All implementing classes are responsible to generate their respective parts of the explanation.

Control of the explanation generation process, and the storing of the explanation data is performed in the ExplanationEngine singleton. While this class does not implement the Explanation interface since it require multiple specialized operations to generated the desired explanation (case) and explanation type (transparency, justification). The class have separate arrays to keep track of the explanation objects tied to each retrieved case.

The ExchangeRaceExplanation with the help of Exchange keeps track of the unit exchanges when case race and player race don't match. ExchangeRaceExplanation contains an array with exchange objects as well as the case identification, while each Exchange object knows which units where exchanged and the similarity between them.

Since the similarity calculation is so complex as it is, and is divided in two main classes (with several auxiliary classes), are the explanation component responsible for keeping track of those changes also split in two: CaseExplanation and ArmyUnitExplanation. CaseExplanation stores all the similarities and data associated with the query, except that which is associated with any units in the query, this is handled by the ArmyUnitSimilarity class.

Changes made during adaptation are based around the set of possible rule violations as described in 4.1.3 and the response to those violations. The Actions enumeration class contains all the possible responses, while the AdaptionRule class maps those response actions to their natural language equalivient.

To map the actions taken when a rule violation is encountered where the Action class created, this records: the violation, the set of actions performed and the unit the action were performed on.

Finally there is the AdaptionExplanation class which is used to generate an explanation based on the data within it. As a method to distinguish the explanations each object is keyed with the case identifier.

This application supports two of the explanation goals discussed by Sørmo et al. (2005): transparency and justification. Transparency is supported by the CaseExplanation branch in Figure 4.8 and partly by the ExchangeRaceExplanation branch, if applicable. The main transparency explanation, displays some informative text in conjunction with all the major calculated similarity values and their equations; in an effort to convey to the user how a case were ranked and subsequently selected by the kNN algorithm. Justification explanations are supported through the AdaptionExplanation- and ExchangeRaceExplanation branches. The ExchangeRaceExplanation is a partial

merger between transparency and justification, but is mostly a justification component. A justification explanation first displays the results of the `ExchangeRaceExplanation` (if applicable) which contains informative information about how the exchange process works and which units where exchanged along with their similarity. The remaining part of the justification explanation comes from the `AdaptionExplanation` component and consist of the set of rules which were violated by the army during the adaption phase and what actions were taken to rectify those violations.

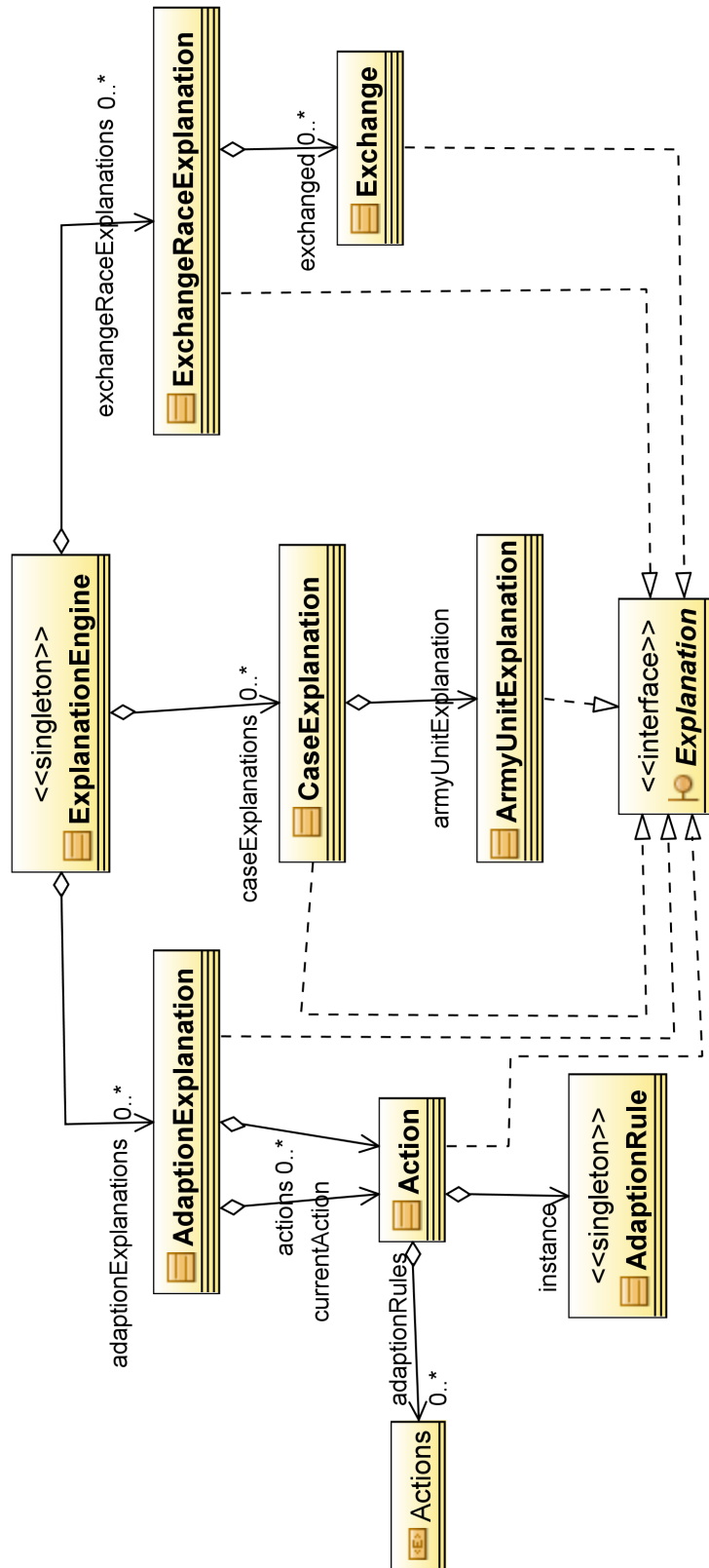


Figure 4.8: Explanation class diagram

4.3 Warhammer and case representation

The representation of Warhammer related classes and how they are connected to cases are very similar to how it is represented in the database; and the representation is displayed in the class diagram in Figure 4.9. To maintain readability of the diagram is all attributes and operations hidden.

The Unit- and UtilityUnit classes inherits from the CoreUnit class which contains most of the similar components for the two generalized classes. Unit is the representation of a Warhammer unit, regardless of unit- and army type, a unit contains absolute knowledge of itself; from characteristics, name, race and cost to what equipment/utility unit it can purchase. It was decided to not include the standard equipment of a unit into the model, this knowledge adds little to the unit representation; and by including the main weapon type of the unit is the weapon classification of the unit maintained. A UtilityUnit is first and foremost an additional unit belonging to another unit, or can be purchased by another unit; it is most often either a mount (horse or beast) or a promoted (enhanced) version of the base unit, but may also be crew-members the base unit requires. In most other aspects are the Unit- and UtilityUnit classes identical.

SpecialRule contains only the rule id and rule name, the effect of the special rule is omitted since the effects rarely matter when selecting the army roster. The inclusion of the special rule field is to help a player to remember which special rules are in effect during gameplay, but have no effect at all in the application. Special rules are added to units and equipment and more than one rule may be added to both units and equipment.

Equipment is the representation of e.g.: weapons, magical items, armor and standards. All items have absolute knowledge about themselves and include the items characteristics and/or bonus(es) when used.

The ArmyUnit class is the first of three classes that are directly associated with a case. It contains which unit, how many units the formation consists of, and the purchased equipment and/or utility units. A key reason to have this class in the representation is to allow all unit knowledge to be contained within one class hierarchy, this is especially linked to the possibility to let units have control of what they can equip, and then the army unit what this particular unit is equipping in this case. It also ensures that units in the database is represented only once, but the unique ID (unit name) is used to link it to all the instances it is used.

An army contains which army units the army consists of along with the available army points and player race. By separating this in its own class/table can one army be used in several cases without the need to represent the date multiple times.

Finally the case is the top most class in the hierarchy, it contains which army is used against which opponent race and the final battle outcome. When a case is loaded from the case base is all the knowledge about the case represented within it. This were a primary concern during design in order to minimize the number of multiple loads of the same data from the database; as well as remove the need to access several hierarchies to acquire all the data.

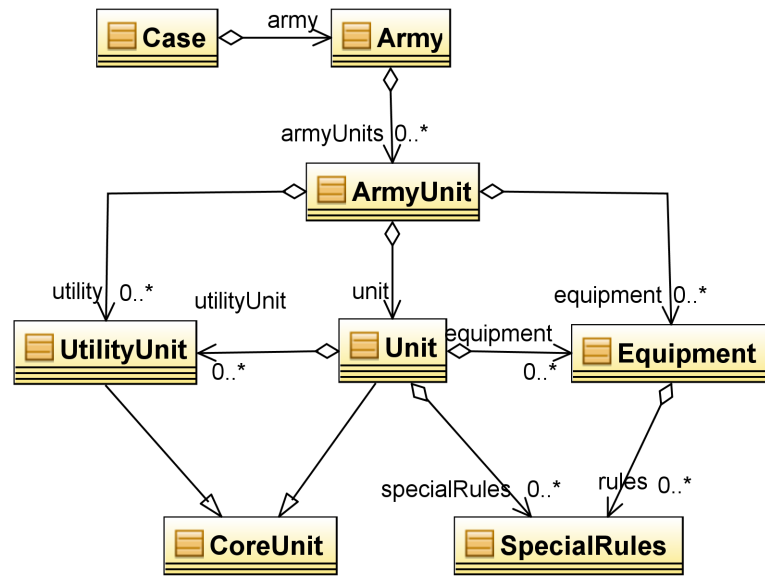


Figure 4.9: Warhammer and case class diagram

Chapter 5

Testing and results

Since the preliminary domain knowledge and cases available to the system is incomplete, is performance testing difficult to perform. Each component and function have however been tested by itself and as a whole, unfortunately is such testing difficult to accurately report. This section will cover some of the testing which have been performed and in Section 6.4 is it mentioned what kind of full scale test the system should be put through when completed. All detailed test results are available in Appendix B.

5.1 Unit weight similarity

One type of tests which have been conducted is tests concerning the unit similarity weights. In order to maximize the unit similarity calculations and adaption matching results the associated weights should be set accordingly. The tests were conducted by calculating two relatively dissimilar units, within the same race, with every possible weight configuration; the weight configuration were in turn only valid within the [0.1-1.0] interval with an increment of 0.1 for each iteration which gives each weight ten possible values. The ten possible values for the six different weights gives a total of $10^6 = 1\ 000\ 000$ possible permutations.

The results of these calculations are not very surprising, since the calculations is based on different features of the unit; it is reasonable to believe, that feature values which by themselves reduce the similarity, will be given no weight by any automated calculations. While on the other hand features which adds to the similarity is given much weight. The end result of these calculations were that the weights are a mixture of 0.1 and 1.0 values.

After the “best” weight permutations had been found these weights were used to calculate the similarity between one unit in a race and all the other race units. The results were then evaluated as either satisfactory (OK) or not (Fail) and the ratio between the two were calculated. This ratio can give a hint towards the overall applicability of the weights.

The results of these test reveals that one universal unit similarity weight configuration

is difficult if not close to impossible to find. This is based on the observations that what is “good” for some units are far from accurate with other units and especially races. As a middle ground could a default weight configuration be calculated and associated with each race. Because of these results were the unit similarity weights locked into the default value of 1.0 for all weights until more extensive testing and better results is collected. Another possible solution to this problem is to rework the entire unit similarity equation, and create something that is better able to distinguish between units.

A similar testing scheme were considered for the case similarity weights, this is however reserved for future work, when more cases and consequently better results are possible. Even though the unit similarity testing did not result in a “good” universal weight configuration do I believe that a similar approach on the cases (1) is a good idea to test the similarity equations and (2) will provide better results.

The result of these tests are presented in Appendix B.1.

5.2 Case adaption

The case adaption stage is a very complex function with several random elements involved, which makes it unpredictable and difficult to control. A specialized test were conducted; in order to verify that neither the random elements involved in the adaption process nor the rule set verification, causes errors. The rule set verification in essentially an infinite loop which can only be broken if no rule violations are detected. This specialized test posted a static query to the system which returned five cases and sent each case through the adaptation 10 000 times totaling 50 000 cases through the adaption engine. Several small bugs were fixed during these tests, these bugs were mostly concerned with deadlocks in some of the automated rule violation procedures, but also some associated with the random elements. The random elements most often emerged after some thousand iterations.

After the successful resolution of the adaption engine errors some tests were conducted in order to verify the soundness of the adapted cases. The results of these tests indicate that in cases which required much adaption (typically cases with few army points being used as the basis for cases with many army points), are prone to adding the same unit several time. This is obviously the result of searching for the most similar unit, but a countermeasure for this incident were included in the process. The countermeasure is supposed to get the next most similar unit if the most similar unit is already used, unless all eligible units are used at least once. Since the conducted tests reveal that the adaption process have an affinity for certain units, is the countermeasure flawed or does not function at all. This type of unit affinity impacts the diversity of the army and may impair the armies overall effectiveness.

Another problem area which emerged during testing is when the lords and/or heroes uses more than 25% percent of the points. These offending lords/heroes have a tendency to loose all their purchased equipment, so it is clear that this part of the adaption should be redesigned.

Based on the results of these test and the amount of work sometimes required to

adapt the case is it likely that the entire adaption logic should be re-implemented and made more intelligent. In the current implementation is it possible for a change made to fix an rule violation, can trigger another rule violation. The adaption engine may as a result swap between several problems until it finds an acceptable configuration.

The result of the test were the unit affinity along with few lord/hero equipment choices becomes apparent are presented in Appendix B.2.

5.3 Race exchange

An interesting application feature is its ability to create a new army from the ground, by basing it on an existing army from another race. The functionality were developed to act in the event that the one or more of the kNN results are of another race than the target race. This event can occur if: there are no cases with the target race in the case-base; the existing cases in the case-base are less similar than other cases with anther race; or if k is larger than the number of cases with the target race. It is interesting to test this feature both to verify that it performs as expected and to verify the soundness of the created army.

For this purpose were Dwarf cases withheld until the very end of the project; by querying for non-existent dwarf cases would the system be forced to create dwarf armies based on unrelated data. No apparent errors were found as a result of the exchange itself, but the same countermeasure which should prevent many duplication units during regular adaption is also present here. The functionality which checks if the original unit had full command status; if the replacement unit is eligible for full command and then apply full command is working flawlessly. Since the exchange race functionality is rather stupid is the regular adaption process required in the majority of the exchanged races to apply rule violation fixes. This is preferable to making the exchange functionality more intelligent as it will introduce new complexity and multiple possible error sources.

The result of one of these tests are presented in Appendix B.3.

5.4 Generated explanations

There are three possible explanations which can be generated during the CBR-cycle and each of them should be used to (1) convey useful information (2) in a structured manner. Each explanation were generated in turn and the output studied to evaluate these two goals.

The best explanation is the one created to explain the race exchange process. The information presented is a merger of both transparency and justification and it is structured, readable and understandable.

Transparency explanations contains all the necessary data to inform the user of how a case were found and classified as similar. It could however been better structured to ease readability.

Justification explanations have a bad layout and poor readability, this is partially because it tries to convey much information, and the adaption process is far from optimal. With a better adaption process (which requires fewer redundant operations) would the explanations be simpler. This could by itself result in better understandability and readability of the explanation. This explanation type would probably be better served with a specialized user interface than the plain text representation it currently posses.

The result of this test are presented in Appendix B.4.

Chapter 6

Discussion

In this chapter will a discussion concerning the two major components of the system (CBR and explanations) be presented. The discussion may include *why* various design decisions were taken, what options were considered, and/or results of the decisions.

In addition to the major components will this chapter discuss the project goals, as described in Section 1.2; and a list of future work which should be done before a fully functional system is made available for the end users.

Section 6.1 discusses CBR in general and each step in the cycle respectively. While explanations in relation to explanation goals are covered in Section 6.2, an evaluation of the project goals is presented in Section 6.3. The chapter is concluded in Section 6.4, where ideas for future work is described.

6.1 Case-based reasoning

CBR are widely used in domains with sparse knowledge or where the knowledge is difficult to translate into rules. Examples of CBR systems are: CARMA (a hybrid of CBR and Model Based Reasoning for rangeland grasshopper control, Branting et al. (1999)) and CASEY (a medical diagnostics tool, Koton (1988)) In this particular domain are the domain well understood, the rules however can be classified as both simple and complex. The standard rules are quite simple and easy to translate into a small rule-set, this small rule-set is the one most likely to be used by the application. However each race and many units have special rules or alterations to the general rule-set that can make it all quite complex, for this reason is most of the class specific rules dropped and the application only focuses on the general rules for army creation. Another component in the decision to use CBR is that the knowledge of armies, units and battles can be viewed as a case without any difficulties. Finally CBR offers the possibility to learn and get more accurate predictions over time, this is a huge benefit over rule-based systems which also were considered. For a full overview of what artificial intelligence approaches were considered the diligent reader is encouraged to read the discussion- and/or technology review chapters in my specialization project (Strandbråten, 2010).

Section 6.1.1 discusses the extent of the conceptualized knowledge in the domain model, which cases were included and why. The four steps in the CBR cycle is discussed in Sections 6.1.2, 6.1.3, 6.1.4 and 6.1.5 respectively.

6.1.1 The case-base

The contents of the knowledge- and case-bases are incomplete, only a few number of races and cases (armies) have been modeled into them. There are several reasons for why much of the knowledge is missing. One reason is that we (my supervisor and me), through discussions realized that in order to begin the development of the system and implement the core functionality; only a small set of data were needed. A unique rulebook is required for each race to be modeled and only a small set of such rulebooks were easily available to us. The Warhammer universe is under constant evolution and many of the races are currently only available for previous versions of the game. The current main rulebook is of the 8th edition, while much of the army books still are 6th or 7th editions. As a result were I reluctant to add too much information as it most probably would result in much redundant work to update the information when the new editions of the rulebooks is released. Finally the initial cases span only a small set of races, although more than those modeled. The cases selected for modeling is based on which races whose rulebooks were available and also which races were best represented in the acquired armies.

At current there are three races fully modeled into the domain model (Empire, Dwarfs and High Elves), additionally is all the core items (e.g.: magical weapons and armor, enchanted items and battle standards), which is available for all races included in the domain model. None of the included races are 8th edition books¹ and will probably require an update when new editions arrive, but they represent those races present in most cases. There are ten cases in the case-base at present (2 Empire-, 5 High Elves- and 3 Dwarf cases) spread between 8 armies.

It could be argued that the above number of cases is insufficient to create an accurate and reliable CBR system, and I agree. However the cases added to the case-base reflects the currently available domain knowledge and represents those races which were most represented in the acquired data. But most importantly; the data is sufficient, and in some cases preferable, when testing the core CBR functionality step-by-step.

6.1.2 Retrieve

In a standard CBR system will all cases in the case-base be retrieved when a query is submitted, before that systems selection scheme extracts the subset of cases to be used. During system conceptualization in the specialization project was it decided that system created cases. with no known outcome (e.g.: have not yet been played) were to be excluded from this initial retrieval. This is to prevent unknown and potentially

¹As of May 2011 are only two of the races updated to 8th editions: *Orcs & Goblins* and *Tomb Kings*

erroneous data to be used as the foundation for new cases. To achieve this a separate database or tables were envisioned.

A better and less labor intensive approach is to tag new cases with *unknown* and then prevent retrieval of cases with that tag. The tag can be envisioned as the key to a safe and all cases tagged with a specific value (unknown) is located inside that safe. This approach reduces the number of copy/delete operations in the database since there is only one field which must be updated, in order to remove the case from the protection of the safe, and give the CBR engine access to it. The old approach with two separate databases would have required a complete data copy of the case from one database/table to another.

The similarity equations described in Section 4.1.2 evolved over time into what they are today. In early builds of the system (with a database model as described in Section 3.1.1) a simple similarity equation were used. The very first similarity were based on the the army points, player- and opponent races and the outcome in a weighted average. An interval value (like today, Equation 4.5) were used on the points, while an enum distance equation were responsible for the three remaining feature similarity calculations (player race, opponent race and outcome). The next evolution of the similarity included a way to check if a query unit is present in the case, but only based on the unit name. Equipment/Utility units and unit numbers where not part of this simple calculation. These similarity equations were ultimately scrapped in order to accommodate the change made to the domain model, although some core ideas from the original similarity equation survived.

When creating a CBR system or an expert system in general it can often be as difficult to get the domain expert to accurately translate his/hers knowledge into rules as to create the knowledge to be contained in the domain model. Another facet of this problem is to create reliable, efficient and correct similarity measures as is implied with the above discussion of the similarity measure evolution. The users of Myrmidia is given complete control over how variables in the case similarity calculations are performed. Through a dedicated compulsory user interface: is the user requested to either approve the default weight configuration along with the army points interval and number of cases to retrieve; or to set their own values. This measure of control is given to the user in the attempt to remove some of the uncertainty concerned with the experts decisions. Figure 4.3 displays this specialized user control UI.

6.1.3 Reuse

The reuse step contains two adaptation techniques *naive* and *exhaustive*. These two techniques emerged naturally during the implementation without much planning in advance. The naive technique arose when changing the retrieved case to reflect the query and where the initial adaption phase. However merely matching the query data is insufficient for the adaption phase, e.g.: telling the case that it should utilize 3000 points, but not giving the program the ability to change the army composition (adding/removing units) is problematic. Thus the exhaustive approach emerged, here are all the rules

verified and associated actions performed when a rule is violated.

As a precaution against queries containing data that by itself violated one or more rule is the exhaustive approach given supreme authority on the adaptation result. This can result in a case which contains none of the queried units, but it was decided that the rules should be the most important factor. Additionally is the adaptation process in its present state constrained enough; in an attempt to simplify the process is the queried units no longer referenced in the exhaustive process. To counter this potential disregard for query units, does the naive and exhaustive approaches work sequentially: First is the naive adaptation performed and those changes have the possibility to trigger the more exhaustive adaptation, if and only if any rule is violated. This results in cases which with high probability contains at least some of the queried units.

6.1.4 Revise

Section 4.1.4 focused on the *how* the revision step of the CBR cycle were implemented, this section will focus on the *why*. During development some consideration went into how this step should be implemented and thoughts ranged from *manual* to its opposite *fully automated* or a mix of the two. In reality would probably even a fully automated step contain the possibility for manual tweaks, but rarely needed. A completely manual process have the advantage that the user is in complete control and the program defaults to “expert” decisions. There is the chance that the user is no expert, but even an accomplished user (and most enthusiastic or even casual gamers are one of the two, at least when it comes to their army/race of choice) will be able to perform the case revision (if needed).

A fully automated approach could have several implementations; as a simulator which acts out a game and ranks the adapted cases based on their score in this simulation. Another approach could be to simply re-rank the adapted cases based on query and other cases (where the case have another player race) which fought successfully against the desired opponent. The simulator approach is far too complex to include alongside everything else and would probably fit better as its own assignment (master thesis), but the more relaxed revise approach could prove useful. In truth an attempt on this were made, but difficulties with the implementation and much other work which also required attention removed it from the production pipeline. Another factor is that only a simple re-ranking between the adapted case and the query would be realistic since the case-base still is far too little to be able to fetch other cases to compare with.

The result of the problems faced, and the realization that only a partial automated revise step could be made; is that the revise step became manual and work on this process could be postponed until after the user interface creation at the end of the implementation period if more time were available.

6.1.5 Retain

Most CBR systems needs a mechanism to manage and delete redundant cases. Smyth and Keane (1995) discusses the *utility problem* and approaches to fix or minimize the problem in CBR application.

The utility problem occurs when the cost associated with searching for relevant knowledge outweighs the benefit of applying this knowledge.
-Smyth and Keane (1995)

In a CBR system is this most often associated with case-bases which grows to an unmanageable size; cases in the case-base may fall under several classifications: *pivotal*, *spanning*, *support* and *auxiliary* cases. Pivotal cases are cases which cannot be reached by any other means, and should only be deleted as a last resort. Spanning cases link (or span) areas in the problem domain that are covered by other individual cases, and deletion of a spanning case may have no negative effect on the systems capability. Support cases are a group of spanning cases, whose deletion by itself or the entire group rarely have a negative effect on the system. Auxiliary cases have absolutely no effect on the systems capabilities, but may reduce the efficiency, and are the first that should be removed

Myrmidia's deletion policy is not too complex nor is it based on a random approach. In order to help maintain performance and capability of the system will all cases designated as a defeat be deleted from the system. The associated army is deleted if and only if it does not exist in another case, either one which is unclassified or a successful case (victory, draw). Cases that fall under this deletion policy may be regarded as *auxiliary* cases.

This approach were selected since a losing case will bring no new and/or advantageous knowledge to the system; although it is true, that an army which previously were defeated, can be victorious against the same or another opponent, at a later date. Another reason is that by regarding defeat cases as undesirable a quick and easy deletion scheme could be utilized.

6.2 Explanation

Section 4.2.1 describes how the explanation framework is built. In this section is some of the reasoning behind that framework described. In order to generate explanations the application must (1) be able to track the system operations as they occur, or (2) be able to reproduce those operations when the explanations is to be generated. There are pros and cons to both approaches; Approach 1 will require more computational power during those operations and more memory to remember them, but it will remain consistent and ensure that everything is stored as it occurs. Approach 2 consumes computational power and memory only when needed, but requires that the same computations is performed multiple times. In addition what happens to the accuracy of the explanation when there are random elements involved?

The first approach seems the most viable, especially since there is a degree of randomness connected with the adaptation process. Another reason for choosing option 1 is the absurd nature of doing the same operations multiple times, even if you could guarantee the accuracy of the operations with the random elements involved.

6.3 Evaluation of goals

This section will evaluate how well this system reaches the goals stipulated in Section 1.2, if at all. Each goal is covered in its own section in the order they were listed (Goal 1-4).

6.3.1 G1: Determine explanation goals

Section 4.2 briefly describes the five explanation goals outlined by Sørmo et al. (2005). Out of these five goals are three selected to be supported by this application: *transparency*, *justification* and *conceptualization*.

Transparency is necessary in order to convey how the system reached its solution. This is usually done by making the data the solution is based upon available for review.

Justification were selected since its primary objective is to increase the users confidence in the solution provided by the system, by explaining why the solution is a “good” solution. By providing this kind of explanation the the users gets an understanding of how the system found the answer, it is also a means of learning and the user may use a similar approach during later manual army creations.

Conceptualization is selected since it can teach the user about terminology used by either the system or the domain (Warhammer). Valid conceptualization questions could be e.g.: “What is a halberd?” (domain question’) or “What is a case?” (system question).

While the three selected explanation goals supplement some learning on their own, teaching is not a goal in itself. For this reason is the learning goal discarded. Relevance, which is mostly used in conversational system as a means to convey the relevance of an asked question is unnecessary.

Three explanation goals for the system were selected during the project conceptualization, but only two of those made in into the product. In my opinion is the generated transparency explanations the best explanation the system can produce. They give an accurate overview of how the process functions; which calculations are performed; and the result of those calculations. The mix of transparency and justification in the explanations generated when the entire army (race) is exchanged is also good and provide detailed information about how the process works and the similarities between the exchanged units. The resulting justification explanations could use a lot more work. While the justification explanation functionality is poorly constructed there is also a concern that as a result of this poor implementation some unit/action mappings might be incorrect. Finally the missing explanation goal *conceptualization*; this goal where not

scrapped or neglected, but postponed since it was decided to use the remaining time to construct a rough user interface to simplify functionality testing and usage.

6.3.2 G2: Create the domain model

The process of developing the domain model is thoroughly described in Section 3.1 and will not be further discussed here. When filling the model with data, and later when that data is accessed and utilized in the application some shortcomings were discovered. The modeling of equipment especially suffers from poorly understood and conceptualized requirements. The equipment cost should have been separated from the equipment, as many items are used or can be used by different units in different races at different costs. This results in multiple table entries which makes the domain model harder to maintain and consequently is the table no longer adhering to the *third normal form* (3NF).

Since there are some problems with the domain this goal was not completely reached, but it is close, and thus considered fulfilled. Additionally this problem is mentioned as suggested work to be performed in future work in Section 6.4.

6.3.3 G3: System creation

Throughout this document have the design (Chapter 3), implementation (Chapter 4) and testing (Chapter 5) of the system been introduced. Collectively do these chapters help define the system and how it was created, and in essence fulfill the requirements for goal 3.

From the specialization project an architectural overview has been available with a clear indication of what is to be made, and the tools to use. Minor modifications and shifts of focus have been conducted after the specialization project was completed, in order to meet the goals and emerging concerns. The design diagrams were created after the completion of the implementation through the use of backwards engineering.

Even though a design was lacking during implementation I feel confident that a good system has been made with structured class hierarchies and task dedicated classes. A full API² has been generated for this system and is included as part of the digital documentation³ of the system.

6.3.4 G4: Evaluation

The final system evaluation is performed in this section, it will be based partly on the results of the test performed in Chapter 5 and partly on the evaluation of the three previous goals (Sections 6.3.1 to 6.3.3). Since it is premature to conduct a full system

²The API is also available at my website: <http://www.seperothproductions.com/Scholar/Myrmidia/API/>

³Digital documentation and deliveries available through DAIM: <https://daim.idi.ntnu.no/>

test; with associated battles and outcome recordings, will such test and/or elaborations on the subject not be part of this evaluation.

Testing conducted on the unit similarity proves the complexity of the domain and the necessity for a AI/CBR system instead of an statistical approach. The statistical approach would have had the same calculation difficulties, but would in addition been more static in its final configuration. A CBR system by its very nature is dynamic and will grow more competent as relevant data are added to its knowledge; which is also one of the major reasons a rule-based system where discarded in the specialization project.

The remaining test also revealed a series of weaknesses which must be addressed in order to achieve high efficiency, performance and reliability. A suggestion on what the remaining work is, are presented in Section 6.4.

Finally is it my opinion that the system have achieved all of its set goals, since each goal is rather lofty and have much room for interpretation. However; while each goal is reached, are not all of them achieved to the best of their potential. Much work is still needed before a truly helpful decision support system is created and deployed.

6.4 Future work

Even though a substantial amount of work have been performed during this semester, is the application as a whole, still in its infancy. The primary focus in this assignment have been to create the foundations for the entire application, and reserve much of the specialization work or “fine tuning” for future work. In this section I propose an unranked non-exhaustive list of ideas on how to build upon and/or improve the current work.

- 1: Improve the knowledge in the case-base, especially the amount of knowledge by modeling even more races and improved cases. Another is to change how equipment is modeled in the database, most prominent is to split the equipment cost into a separate table. The need for this became apparent late in the development when the same item had to be added many times in the table to accommodate the different costs, e.g.: Musician and Standard bearers are modeled six or seven times each.

- 2: Improve the adaption process in order to reduce the number of needed operations. It was also discovered during testing that the adaption engine favors single units above others, even though measures where taken to prevent this e.g.: *Captains of the Empire* and *Halberdiers* are preferred over other Empire units and appear often when much adaption is needed. This problem may be improved by supplying better cases and knowledge which will reduce the number of needed adaptations.

- 3: Improve the user interface, the user interface provided is quickly implemented and poorly executed since the focus lay elsewhere in this assignment.

- 4: Implement the explanation goal *conceptualization*. This is one of the planned explanation types, but time did not suffice.

- 5: Implement a partially automated revise step, which could include e.g.: re-ranking of the adapted cases and suggestions on how to further improve the case before deployment.

6: After adding more data to the domain model and case-base should an extensive test/verification of the system be performed. This is not done now since the program currently is very restricted in applicable player races. The test could be performed as a tournament in association with a Warhammer club and could be executed in one of two ways depending on what statistics are desirable to collect. (1) All armies are created by this system, and then fight battles with the created armies. This could be used to verify the soundness of the created armies as well as the integrity of the cases present in the case-base. (2) Split the army creation process between hand-crafted and system-crafted armies and arrange the matches so that the two types fight each other. The results from this approach could verify the performance of the system and if the cases created are fit for the task they were created.

7: Perform calculations on the case similarity weights to improve the matching between query-case and/or case-case. Inspiration for this test can be drawn from the unit similarity test performed and detailed in Section 5.1 and Appendix B.1.

Chapter 7

Conclusion

In this thesis have I presented an explanation aware decision support system based on CBR technology. This system is designed to help Warhammer gamers to create the army they should use against a specified foe. The work have primarily been focused on creating all the necessary components the system need and then weave them together, into the first system iteration.

This CBR system uses knowledge of previously successful battles in order to make its recommendation, these battles are supplied by professional hobby gamers at the local Warhammer community (WarTrond). Through data recorded during retrieval and adaption stages are justification and transparency explanations generated; these explanations accurately describes the reasoning behind the recommendation, but could be presented in a better manner.

The most prominent problem found during testing is the adaption process' affinity towards utilizing a selected few units during case/army adaption. This affinity reduces the: versatility of the army, the effectiveness and the probability of victory.

It is my opinion that the resulting program of this master thesis, supply a great foundation and build block for future master students which desires to continue this work. Much work is still needed in order to create a truly helpful decision support system, which can be deployed and used by Warhammer gamers worldwide.

Bibliography

- Aamodt, A. (1994). Explanation-Driven Case-Based Reasoning. *In: S. Wess, K. Althoff, M. Richter (eds): Topics in case-based reasoning*, pages 274–288.
- Aamodt, A. and Plaza, E. (1994). Case-based Reasoning; Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, 7(1):39–59.
- Apache Software Foundation (2010). *Getting started with Derby*. Apache Software Foundation. <http://db.apache.org/derby/docs/10.7/getstart/getstartderby.pdf>, Last accessed: 2011-04-06.
- Bernard, E., Ebersole, S., and King, G. (2011). Hibernate EntityManager User Guide. Technical report, hibernate.org. http://docs.jboss.org/hibernate/entitymanager/3.6/reference/en/pdf/hibernate_reference.pdf Last Accessed: 4 May, 2011.
- Branting, L. K., Hastings, J. D., and Lockwood, J. A. (1999). Integrating Cases and Models for Prediction in Biological Systems. *AI Applications*, 11:29–48.
- Díaz-Agudo, B. and González-Calero, P. A. (2001). A Declarative Similarity Framework for Knowledge Intensive CBR. In *Int. Conf. on Case-Based Reasoning ICCBR-2001 (2001)*, pages 158–172. Springer.
- Doyle, D., Tsymbal, A., and Cunningham, P. (2003). A review of explanation and explanation in case-based reasoning. Technical report, Department of computer Science. Trinity.
- Games Workshop (2009). *Warhammer Rulebook*. Games Workshop, 8th edition.
- Garcia, J. A. R., Diaz-Agudo, B., and González-Calero, P. (2008). jCOLIBRI 2 Tutorial. Technical report, Group for Artificial Intelligence Applications Universidad Complutense De Madrid.
- Hinrichs, T. R. (1992). *Problem Solving in Open Worlds: A Case Study in Design*. Psychology Press.

- Kass, A. M. and Leake, D. B. (1988). Case-Based Reasoning Applied to Constructing Explanations. In Kolodner, J., editor, *Proceedings of 1988 Workshop on Case-Based Reasoning*, pages 190–208. Morgan Kaufmann.
- Kofod-Petersen, A. and Aamodt, A. (2009). Case-based Reasoning for Situation-aware Ambient Intelligence: A Hospital Ward Evaluation Study.
- Koton, P. (1988). Reasoning about evidence in causal explanations. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88)*, pages 256–261. AAAI Press.
- Leake, D. B. (1995a). Abduction, Experience, and Goals: A Model of Everyday Abductive Explanation. *JOURNAL OF EXPERIMENTAL AND THEORETICAL ARTIFICIAL INTELLIGENCE*, 7:407–428.
- Leake, D. B. (1995b). Goal-Based Explanation Evaluation. In *Goal-Driven Learning*, pages 251–285. MIT Press.
- Richter, M. M. (1995). The knowledge contained in similarity measures. Invited Talk at the First International Conference on Case-Based Reasoning, ICCBR'95, Sesimbra, Portugal.
- Riesbeck, C. K. and Schank, R. C. (1989). *Inside Case-Based Reasoning*. Erlbaum.
- Roth-Berghofer, T. (2004). Explanations and Case-Based Reasoning: Foundational Issues. In Funk, P. and Calero, P. A. G., editors, *Proceedings of the 7th European Conference on Case-based Reasoning (ECCBR 2004)*, volume 3155 of *Lecture Notes in Computer Science*, pages 389–403. Springer.
- Roth-Berghofer, T. R. and Cassens, J. (2005). Mapping Goals and Kinds of Explanations to the Knowledge Containers of Case-Based Reasoning Systems. In *Proceedings ICCBR 2005*, pages 451–464. Springer.
- Schank, R. C. (1986). *Explanation Patterns: Understanding Mechanically and Creatively*. Lawrence Erlbaum Associates.
- Smyth, B. and Keane, M. T. (1995). Remembering To Forget: A Competence-Preserving Case Deletion Policy for Case-Based Reasoning Systems. pages 377–382. Morgan Kaufmann.
- Sørmo, F., Cassens, J., and Aamodt, A. (2005). Explanation in Case-Based Reasoning: Perspectives and Goals. *Artificial Intelligence Review*, 24(2):109–143.
- Strandbråten, G. R. (2010). Warhammer Fantasy Battle Army Builder. TDT4500 - Specialization Project at NTNU, Department of Computer and Information Science, group for Artificial Intelligence.

Watson, I. (1998). CBR is a methodology not a technology. In *Research & Development in Expert Systems XV*, pages 213–223. Springer, London.

Appendix A

Glossary

| | |
|--------------|---|
| 3NF | Third normal form, database normalization law |
| A | Attacks (Related to: Warhammer) |
| A posteriori | Knowledge found through experience/knowledge known in hindsight |
| A priori | Knowledge known in advance |
| AI | Artificial Intelligence |
| Aka | Also known as |
| AoE | Area of Effect |
| API | Application Programmer Interface |
| BS | Ballistic skill (Related to: Warhammer) |
| BSB | Battle Standard Bearer (Related to: Warhammer) |
| Ca | Cavalry (Related to: Warhammer) |
| CBR | Case-Based Reasoning |
| Ch | Chariots (Related to: Warhammer) |
| D&D | Dungeons and Dragons |
| E.g. | Exempli gratia: For the sake of example (in English usually shortened to “for example”) |
| Et al. | Et alii: And others |
| Framework | Is an abstraction in which common code providing generic functionality, can be selectively overridden or specialized by user code |
| Full Command | Term used to denote that a unit/formation have an musician, standard bearer and a promoted unit(Related to: Warhammer) |
| HQL | Hibernate Query Language |
| I.e. | Id est: “that is (to say)”/“in other words”/“which means” |

| | |
|--------------|--|
| In | Infantry (Related to: Warhammer) |
| I | Initiative (Related to: Warhammer) |
| Ld | Leadership (Related to: Warhammer) |
| JDBC | Java Database Connectivity |
| JVM | Java Virtual Machine |
| kNN | k-NearestNeighbour |
| M | Movement Allowance (Related to: Warhammer) |
| Mo | Monsters (Related to: Warhammer) |
| MB | Monstrous Beasts (Related to: Warhammer) |
| MC | Monstrous Cavalry (Related to: Warhammer) |
| MI | Monstrous Infantry (Related to: Warhammer) |
| MMORPG | Massively Multi player Online Role-Playing Game |
| NTNU | Norwegian University of Science and Technology |
| POJO | Plain Old Java Object |
| RBS | Rule-Based Systems |
| RDBMS | Relational Database Management System |
| S | Strength (Related to: Warhammer) |
| SQL | Simple Query Language, language used to query data from a database |
| Sw | Swarms (Related to: Warhammer) |
| T | Toughness (Related to: Warhammer) |
| (G)UI | (Graphical) User Interface |
| UML | Uniformed Modeling Language |
| Un | Unique Units (Related to: Warhammer) |
| Utility Unit | Is the classification given to mounts, crewmen and units which are given a promotion that regular “units” can be improved with (Related to: Warhammer) |
| Vice versa | And the other way around |
| vs. | Versus, Used to link two or more opposing or contrasting elements |
| W | Wounds (Related to: Warhammer) |
| WB | War Beasts (Related to: Warhammer) |
| WFB | Warhammer Fantasy Battle |
| WM | War Machines (Related to: Warhammer) |
| WS | Weapon skill (Related to: Warhammer) |

Appendix B

Test results

In this appendix is all the test result data listed in their associated tables along with some explanation of the results and how they were found. Some of this information will be similar to that of Chapter 5.

B.1 Unit weights

Tables B.1 and B.2 displays the weights calculated as the best possible permutation based on similarities between two units. The results is as expected since all features with large differences will reduce the similarity while similar features will count more.

Table B.1: Unit similarity weight, Test: 1

| Empire Archers vs. Empire Steam Tank | |
|--------------------------------------|-------|
| Weight | Value |
| Characteristics | 0.1 |
| Unit type | 0.1 |
| Army type | 1.0 |
| Weapon type | 0.1 |
| Cost | 1.0 |
| Magician | 1.0 |

Table B.2: Unit similarity weight, Test: 2

| Empire Emperor Karl Franz vs. Empire Steam Tank | |
|---|--------------|
| Weight | Value |
| Characteristics | 0.1 |
| Unit type | 0.1 |
| Army type | 1.0 |
| Weapon type | 0.1 |
| Cost | 0.1 |
| Magician | 1.0 |

A third similarity test where also conducted between *Emperor Karl Franz* and *Archers*, the results of this comparison however, where identical to the results in Table B.2 and is therefore not displayed in its own table.

To increase the result confidence, were a random unit (not one of the units used during the weight calculation) selected as the control unit. The control unit were then compared to each unit in the target race using the previously calculated weights as well as the default weights (all weights set to 1.0). This approach were used in the three races present in the database and the results are summarized in Tables B.3, B.4 and B.5. Each of the unit similarities calculated by using the two *best* weight configurations were then evaluated as either *OK* or *Fail* based on how well the similarities matches usage areas and to a certain degree the default similarities. Finally will the *OK/Fail ratio* give a pointer to the applicability of the weight configuration, the higher the ratio the better the configuration is.

Table B.3: Unit similarity test: Empire

| Race | | Empire | | | |
|----------------------------|---------|---------------------------------------|------------|----------|------------|
| Control unit | | Halberdier | | | |
| Weight 1 | | Unit similarity weight, Test: 1 (B.1) | | | |
| Weight 2 | | Unit similarity weight. Test: 2 (B.2) | | | |
| Unit name | Default | Weight 1 | Evaluation | Weight 2 | Evaluation |
| Arch Lector Of Sigmar | 0.383 | 0.208 | OK | 0.153 | OK |
| Archers | 0.667 | 0.542 | OK | 0.867 | Fail |
| Balthasar Gelt | 0.383 | 0.208 | OK | 0.153 | OK |
| Battle Wizard | 0.383 | 0.208 | OK | 0.153 | OK |
| Captain of the Empire | 0.55 | 0.625 | OK | 0.82 | Fail |
| Crossbowmen | 0.667 | 0.542 | OK | 0.867 | Fail |
| Emperor Karl Franz | 0.476 | 0.606 | Fail | 0.79 | Fail |
| Flagellant Warband | 0.531 | 0.62 | OK | 0.813 | Fail |
| Free Company Fighter | 0.883 | 0.708 | Fail | 0.953 | Fail |
| General of the Empire | 0.55 | 0.625 | Fail | 0.82 | Fail |
| Grand Theogonist Volkmar | 0.5 | 0.5 | Fail | 0.2 | OK |
| Great Cannon | 0.204 | 0.426 | OK | 0.681 | Fail |
| Greatswords | 0.592 | 0.729 | OK | 0.837 | OK |
| Halberdiers | 1 | 1 | OK | 1 | OK |
| Handgunners | 0.667 | 0.542 | OK | 0.867 | Fail |
| Helblaster Volley Gun | 0.204 | 0.426 | OK | 0.681 | Fail |
| Helstorm Rocket Battery | 0.204 | 0.426 | OK | 0.681 | Fail |
| Knight of the Inner Circle | 0.383 | 0.583 | OK | 0.753 | Fail |
| Knights | 0.55 | 0.625 | Fail | 0.82 | Fail |
| Kurt Helborg | 0.494 | 0.611 | Fail | 0.798 | Fail |
| Ludwig Schwarzhelm | 0.365 | 0.579 | Fail | 0.746 | Fail |
| Luthor Huss | 0.425 | 0.688 | Fail | 0.77 | Fail |
| Master Engineer | 0.55 | 0.625 | Fail | 0.82 | Fail |
| Mortar | 0.204 | 0.426 | OK | 0.681 | Fail |
| Outriders | 0.333 | 0.458 | OK | 0.733 | Fail |
| Pistoliers | 0.333 | 0.458 | OK | 0.733 | Fail |
| Spearmen | 1 | 1 | OK | 1 | OK |
| Steam Tank | 0.185 | 0.421 | Fail | 0.674 | Fail |
| Swordsmen | 0.883 | 0.708 | OK | 0.953 | OK |
| Templar Grand Master | 0.328 | 0.569 | Fail | 0.731 | Fail |
| Warrior Priest | 0.55 | 0.625 | Fail | 0.82 | Fail |
| Wizard Lord | 0.383 | 0.208 | OK | 0.153 | OK |
| OK/Fail ratio | | | 1.667 | | 0.391 |

Table B.4: Unit similarity test: Dwarfs

| Race | | Dwarfs | | | |
|-----------------|---------|---------------------------------------|------------|----------|------------|
| Control unit | | Dwarf warrior | | | |
| Weight 1 | | Unit similarity weight. Test: 1 (B.1) | | | |
| Weight 2 | | Unit similarity weight. Test: 2 (B.2) | | | |
| Unit name | Default | Weight 1 | Evaluation | Weight 2 | Evaluation |
| Bolt Thrower | 0.389 | 0.847 | Fail | 0.756 | Fail |
| Cannon | 0.389 | 0.847 | Fail | 0.756 | Fail |
| Daemon Slayer | 0.611 | 0.903 | Fail | 0.844 | Fail |
| Dragon Slayer | 0.667 | 0.917 | Fail | 0.867 | Fail |
| Dwarf Lord | 0.63 | 0.907 | Fail | 0.852 | Fail |
| Dwarf warriors | 1 | 1 | OK | 1 | OK |
| Flame Cannon | 0.389 | 0.847 | Fail | 0.756 | Fail |
| Grudge Thrower | 0.389 | 0.847 | Fail | 0.756 | Fail |
| Gyrocopter | 0.463 | 0.866 | Fail | 0.785 | Fail |
| Hammerers | 0.583 | 0.708 | OK | 0.833 | OK |
| Ironbreaker | 0.667 | 0.917 | OK | 0.867 | OK |
| Longbeards | 0.833 | 0.958 | OK | 0.933 | OK |
| Master Engineer | 0.667 | 0.917 | Fail | 0.867 | Fail |
| Miners | 0.583 | 0.708 | OK | 0.833 | Fail |
| Organ Gun | 0.389 | 0.847 | Fail | 0.756 | Fail |
| Quarrellers | 0.667 | 0.542 | Fail | 0.867 | OK |
| Runelord | 0.667 | 0.917 | Fail | 0.867 | Fail |
| Runesmith | 0.667 | 0.917 | Fail | 0.867 | Fail |
| Slayers | 0.667 | 0.917 | Fail | 0.867 | Fail |
| Thane | 0.667 | 0.917 | Fail | 0.867 | Fail |
| Thunderers | 0.667 | 0.542 | Fail | 0.867 | OK |
| OK/Fail ratio | | | 0.312 | | 0.4 |

Table B.5: Unit similarity test: High Elves

| Race | | High Elves | | | |
|---------------------------|---------|---------------------------------------|------------|----------|------------|
| Control unit | | Spearmen | | | |
| Weight 1 | | Unit similarity weight. Test: 1 (B.1) | | | |
| Weight 2 | | Unit similarity weight. Test: 2 (B.2) | | | |
| Unit name | Default | Weight 1 | Evaluation | Weight 2 | Evaluation |
| Alith Anar | 0.426 | 0.481 | Fail | 0.77 | Fail |
| Archers | 0.833 | 0.583 | OK | 0.933 | Fail |
| Archmage | 0.383 | 0.208 | OK | 0.153 | OK |
| Caradryan | 0.55 | 0.625 | Fail | 0.82 | Fail |
| Dragon Mage of Caledor | 0.383 | 0.208 | OK | 0.153 | OK |
| Dragon Princes of Caledor | 0.383 | 0.583 | Fail | 0.753 | Fail |
| Ellyrian Reavers | 0.5 | 0.875 | Fail | 0.8 | Fail |
| Eltharion | 0.444 | 0.486 | Fail | 0.778 | Fail |
| Great Eagle | 0.346 | 0.574 | Fail | 0.739 | Fail |
| Korhil | 0.55 | 0.625 | Fail | 0.82 | Fail |
| Lion Chariot of Chrace | 0.314 | 0.66 | OK | 0.726 | Fail |
| Lothern Sea Guard | 0.833 | 0.958 | OK | 0.933 | OK |
| Mage | 0.383 | 0.208 | OK | 0.153 | OK |
| Noble | 0.55 | 0.625 | Fail | 0.82 | Fail |
| Phoenix Guard | 0.667 | 0.917 | OK | 0.867 | OK |
| Prince | 0.494 | 0.611 | Fail | 0.798 | Fail |
| Repeater Bolt Thrower | 0.204 | 0.426 | Fail | 0.681 | Fail |
| Shadow Warriors | 0.5 | 0.5 | OK | 0.8 | Fail |
| Silver Helms | 0.383 | 0.583 | OK | 0.753 | OK |
| Spearmen | 1 | 1 | OK | 1 | OK |
| Sword Masters of Hoeth | 0.592 | 0.729 | OK | 0.837 | OK |
| Teclis | 0.383 | 0.208 | OK | 0.153 | OK |
| Tiranoc Chariot | 0.222 | 0.431 | Fail | 0.689 | Fail |
| Tyrion | 0.291 | 0.56 | Fail | 0.716 | Fail |
| White Lions of Chrase | 0.592 | 0.729 | Fail | 0.837 | Fail |
| OK/Fail ratio | | | 0.923 | | 0.562 |

B.2 Adaption result

Table B.6 displays the results of an adaption process on an Empire army. As evidenced by the content of this army is its unit diversity slim. Five formations of Halberdiers totaling a number of 151 Halberdiers and three captains without any equipment and/or assigned as the battle standard bearer. The overwhelming presence of Halberdiers and

the total lack of ranged units with the exception of the war machines makes this army a difficult one to play.

Table B.6: Adaption result - unit preference error

| Unit name | Equipment | Utility unit | Cost |
|--------------------------|---|-----------------|-------------|
| Arch Lector of Sigmar | | | 125 |
| Helblaster Volley Gun | | | 110 |
| 10xFlagellant Warband | | | 100 |
| Captain of the Empire | | | 50 |
| Captain of the Empire | | | 50 |
| 29xHalberdiers | Full command | | 165 |
| 18xGreatswords | Full command | | 210 |
| 18xSwordmasters of Hoeth | Full command Banner of Sorcery | | 350 |
| Helblaster Volle Gun | Full command | | 110 |
| 29xHalberdiers | Full command | | 165 |
| 31xHalberdiers | Full command | | 175 |
| Captain of the Empire | | | 50 |
| Great Cannon | | | 300 |
| Templar Grand Master | Channeling staff The tricksters shard Berserker sword | Barded warhorse | 197 |
| 32xHalberdiers | Full command | | 180 |
| 10xFlagellant Warband | | | 100 |
| 30xHalberdiers | Full command | | 200 |
| 18xGreatswords | Full command | | 210 |
| Total cost: | | | 2497 |

B.3 Race exchange result

One of the results of the exchange race functionality tests are presented below. This test is performed on the dwarven race before any dwarf cases were added to the case-base. The complete textual explanation is not presented here, but the important information is extracted and placed in a Table B.7

Table B.7: Race unit exchange table

| | | |
|------------------------|----------------|------------|
| Player query race | Dwarf | |
| Player case race | High Elves | |
| Opponent query race | High Elves | |
| Opponent case race | High Elves | |
| Query/Case similarity | 88.88% | |
| Outcome | Victory | |
| Case unit | Exchanged unit | Similarity |
| Spearmen | Dwarf warriors | 86.48% |
| Sword Masters of Hoeth | Ironbreakers | 89.81% |
| Noble | Thane | 81.48% |
| Teclis | Runelord | 62.96% |
| Caradryan | Thane | 81.48% |
| Great Eagle | Gyrocopter | 66.66% |
| Spearmen | Dwarf warriors | 86.48% |
| Repeater Bolt Thrower | Organ Gun | 66.66% |
| Korhil | Thane | 81.48% |
| Repeater Bolt Thrower | Organ Gun | 66.66% |
| Great Eagle | Gyrocopter | 66.66% |
| Sword Masters of Hoeth | Ironbreakers | 89.81% |

Other actions were performed during this adaption in order to guarantee that all rules are followed, but those actions are not transcribed here as they are not part of this test. Although the results of this exchange is far from optimal, and most probably would not be accepted by any gamer without some modification; can it be used as the starting position, from which manual changes could be made in contrast to create everything from scratch.

B.4 Explanation prints

This section displays the unformatted explanations as they appear in the application when a user requests them.

The first box contains the Transparency explanation:

Case #6 have a similarity of 0.8888888888888888 with the query.
 Based on the weighted average of three components:
 (ArmySimilarity, OpponentSimilarity and OutcomeSimilarity).

Army Similarity = 0.6666666666666666 and have a weight of 1.0.

Army similarity is based on the weighted average of:
 PlayerRaceSimilarity, ArmyPointSimilarity and ArmyUnitSimilarity
 Army points(2500) in query have a similarity of 1.0 with the
 army points(2500) in the case, and have a weight of 1.0
 Query player race(High_Elves) have a similarity of 0.0 with
 the player race(Dwarfs) in the case, and have a weight
 of: 1.0
 ArmyUnit Similarity = 1
 The query were empty so the assumed ArmyUnit similarity
 is: 1

Opponent Similarity = 1.0 and have a weight of 1.0
 Query opponent race(High_Elves) have a similarity of 1.0 with
 the opponent race(High_Elves) in the case

Outcome Similarity = 1.0 and have a weight of 1.0
 Query outcome(Victory) have a similarity of 1.0 with the
 outcome(Victory) of the case

This gives the similarity calculation:

$$\text{similarity} = (\text{armySimilarity} * \text{armyWeight} + \text{opponentSimilarity} * \text{opponentWeight} + \text{outcomeSimilarity} * \text{outcomeWeight}) / (\text{armyWeight} + \text{opponentWeight} + \text{outcomeWeight})$$

$$\text{similarity} = (0.6666666666666666 * 1.0 + 1.0 * 1.0 + 1.0 * 1.0) / (1.0 + 1.0 + 1.0)$$

$$\text{similarity} = (2.6666666666666665) / 3.0$$

$$\text{similarity} = 0.8888888888888888$$

The second box contains the justification explanation:

The unit/formation: Empire:Handgunners, where changed based on:
 Error: TOO_FEW_POINTS_TOTAL, where the following action(s)
 taken:
 Least_Similar_Unit: Found the least similar unit to the
 units/formations already in the army with the given
 army type. To add the least similar unit is probably
 a good idea, in order to improve the army.
 Added_Full_Command: Added full command to the unit to
 increase point usage and unit efficiency
 The unit/formation: Empire:Balthasar Gelt, where changed based on:
 Error: TOO_FEW_POINTS_TOTAL, where the following action(s) taken:
 Added_General: Added a general since there were no general in
 the army
 The unit/formation: Empire:Great Cannon, where changed based on:

Error: TOO_FEW_POINTS.TOTAL, where the following action(s) taken:
Least_Similar_Unit: Found the least similar unit to the units/formations already in the army with the given army type. To add the least similar unit is probably a good idea, in order to improve the army

The unit/formation: Empire:Mortar, where changed based on:
Error: TOO_FEW_POINTS.TOTAL, where the following action(s) taken:
Least_Similar_Unit: Found the least similar unit to the units/formations already in the army with the given army type. To add the least similar unit is probably a good idea, in order to improve the army

The unit/formation: Empire:Otriders, where changed based on:
Error: TOO_MANY_POINTS.TOTAL, where the following action(s) taken:
Removed_Special_Group: Removed a special group to reduce the total points used, or to reduce the number of special points; since the used special points exceeded the available special points

The unit/formation: Empire:Greatswords, where changed based on:
Error: TOO_FEW_POINTS.TOTAL, where the following action(s) taken:
Added_Full_Command: Added full command to the unit to increase point usage and unit efficiency
Increased_Unit_Size: Increased the unit size, since there were too few units in the formation to meet the minimum size requirement

The unit/formation: Empire:Knightly Orders, where changed based on:
Error: TOO_FEW_POINTS.TOTAL, where the following action(s) taken:
Added_Full_Command: Added full command to the unit to increase point usage and unit efficiency

The unit/formation: Empire:Handgunners, where changed based on:
Error: TOO_FEW_POINTS.TOTAL, where the following action(s) taken:
Added_Full_Command: Added full command to the unit to increase point usage and unit efficiency

The unit/formation: Empire:Crossbowmen, where changed based on:
Error: TOO_FEW_POINTS.TOTAL, where the following action(s) taken:
Added_Full_Command: Added full command to the unit to increase point usage and unit efficiency
Increased_Unit_Size: Increased the unit size, since there were too few units in the formation to meet the minimum size requirement

The unit/formation: Empire:Halberdiers, where changed based on:
Error: TOO_FEW_POINTS.TOTAL, where the following action(s) taken:
Added_Full_Command: Added full command to the unit to increase point usage and unit efficiency
Increased_Unit_Size: Increased the unit size, since there were too few units in the formation to meet the minimum size requirement

The unit/formation: Empire:Mortar, where changed based on:

Error: TOO_MANY_POINTS.TOTAL, where the following action(s) taken:
 Removed_Special_Group: Removed a special group to reduce the total points used, or to reduce the number of special points; since the used special points exceeded the available special points

The unit/formation: Empire:Great Cannon, where changed based on:
 Error: TOO_MANY_POINTS.TOTAL, where the following action(s) taken:
 Removed_Special_Group: Removed a special group to reduce the total points used, or to reduce the number of special points; since the used special points exceeded the available special points

The unit/formation: Empire:Kurt Helborg, where changed based on:
 Error: TOO_FEW_POINTS.TOTAL, where the following action(s) taken:
 Added_General: Added a general since there were no general in the army

The unit/formation: Empire:Balthasar Gelt, where changed based on:
 Error: TOO_MANY_LORD_POINTS, where the following action(s) taken:
 Removed_Character: Removed a character to reduce the total points used

The unit/formation: Empire:Crossbowmen, where changed based on:
 Error: TOO_FEW_POINTS.TOTAL, where the following action(s) taken:
 Least_Similar_Unit: Found the least similar unit to the units/formations already in the army with the given army type. To add the least similar unit is probably a good idea, in order to improve the army
 Increased_Unit_Size: Increased the unit size, since there were too few units in the formation to meet the minimum size requirement

The unit/formation: Empire:Crossbowmen, where changed based on:
 Error: TOO_MANY_POINTS.TOTAL, where the following action(s) taken:
 Decreased_Unit_Size: Decreased formation size to bring it below the maximum formation size limit, or to decrease point usage

The third box contains the Transparency/Justification explanation generated when a race is exchanged:

The army race were changed from High_Elves to Dwarfs, since the case army did not fit the query army. All units were exchanged with the most similar corresponding unit in the new race. The most similar unit is determined based on unit characteristics; similarity tables between Army-, Unit, and Weapon types; the units base cost; and if the unit is a magician or not.

High_Elves:Repeater Bolt Thrower where exchanged with:

Dwarfs:Organ Gun. The similarity between them is: 0.66
High_Elves:Sword Masters of Hoeth where exchanged with:
Dwarfs:Ironbreaker. The similarity between them is: 0.89
High_Elves:Korhil where exchanged with: Dwarfs:Thane.
The similarity between them is: 0.81
High_Elves:Noble where exchanged with: Dwarfs:Thane.
The similarity between them is: 0.81
High_Elves:Great Eagle where exchanged with:
Dwarfs:Gyrocopter. The similarity between them is: 0.66
High_Elves:Great Eagle where exchanged with:
Dwarfs:Gyrocopter. The similarity between them is: 0.66
High_Elves:Teclis where exchanged with: Dwarfs:Runelord.
The similarity between them is: 0.62
High_Elves:Sword Masters of Hoeth where exchanged with:
Dwarfs:Ironbreaker. The similarity between them is: 0.89
High_Elves:Caradryan where exchanged with: Dwarfs:Thane.
The similarity between them is: 0.81
High_Elves:Repeater Bolt Thrower where exchanged with:
Dwarfs:Organ Gun. The similarity between them is: 0.66
High_Elves:Spearmen where exchanged with:
Dwarfs:Dwarf warriors. The similarity between them is: 0.86
High_Elves:Spearmen where exchanged with:
Dwarfs:Dwarf warriors. The similarity between them is: 0.86

Appendix C

Cases

Each of the ten included cases are presented in detail in this appendix, note that while all cases are presented as they appear in the case base they will have varying level of detail. This discrepancy in detail reflects the original source material and also helps to demonstrate the flexibility of the system. Unfortunately does four of the cases result in defeat, this is far from optimal, and as discussed in Section 6.1.5 will all future cases classified as *defeat* be deleted from the case-base. It is however deemed necessary to include these cases in this test case-base in order to increase the number of cases. When the application moves from testing/development to deployment should these cases have been replaced with other more preferable cases classified as either *victory* or *draw*.

Table C.1: The 10 cases in the case-base

| Case ID | Army ID | Army points | Opponent | Outcome |
|---------|---------|-------------|-------------------|---------|
| 1 | A2 | 1500 | Wood Elves | Victory |
| 2 | A1 | 2500 | Brettonnia | Defeat |
| 3 | A5 | 1750 | Dwarfs | Defeat |
| 4 | A3 | 2500 | Daemons of Chaos | Defeat |
| 5 | A4 | 2500 | Warriors of Chaos | Defeat |
| 6 | A3 | 2500 | High Elves | Victory |
| 7 | A3 | 2500 | Warriors of Chaos | Victory |
| 8 | A6 | 2500 | Orcs and Goblins | Victory |
| 9 | A7 | 2500 | Orcs and Golbins | Victory |
| 10 | A8 | 1750 | High Elves | Victory |

Table C.2: Case army A1

| Army ID: A1 - Empire | | | |
|-----------------------------|--|---------------------|-------------|
| Unit name | Equipment | Utility unit | Cost |
| Arch Lector of Sigmar | Mace of Helstrum Armour of Fortune Potion of Speed | | 255 |
| Wizard Lord | Level 4 Rod of Power Opal Amulet | Barded steed | 276 |
| Captain of the Empire | BSB Full plate armour Griffon Standard | | 138 |
| Warrior priest | Armour of Meteoric Iron Two hammers | | 119 |
| Master Engineer | Repeater Pistol | | 75 |
| 38xHalberdiers | Full command | | 210 |
| 20xSpearmen | Shields Full command | | 140 |
| 20xSwordsmen | Full command | | 145 |
| 10xCrossbowmen | | | 80 |
| 10xHandgunners | Repeater Handgun | Marksman | 105 |
| 19xGreatswords | Full command | | 220 |
| Mortar | | | 75 |
| Mortar | | | 75 |
| Great Cannon | | | 100 |
| Great Cannon | | | 100 |
| Helstrom Rocket Battery | | | 115 |
| Steam Tank | | | 300 |
| Total cost: | | | 2498 |

Table C.3: Case army A2

| Army ID: A2 - Empire | | | |
|-----------------------------|------------------|---------------------|-------------|
| Unit name | Equipment | Utility unit | Cost |
| Templar Grand Master | | | 145 |
| Warrior Priest | | | 90 |
| Battle Wizard | Level 2 | | 100 |
| 40xHalberdiers | | | 200 |
| 20xGreatswords | | | 200 |
| 12xKnights | | | 276 |
| 10xHandgunners | | | 80 |
| 10xCrossbowmen | | | 80 |
| 5xOutriders | | | 105 |
| Captain of the Empire | BSB | | 75 |
| Total cost: | | | 1276 |

Table C.4: Case army A3

| Army ID: A3 - High Elves | | | |
|---------------------------------|-----------------------------------|---------------------|-------------|
| Unit name | Equipment | Utility unit | Cost |
| Teclis | | | 475 |
| Noble | BSB Armour of Caledor | | 135 |
| Korhil | | | 140 |
| Caradryan | | | 175 |
| 33xSpearmen | Musician Standard bearer | | 312 |
| 32xSpearmen | Full command | | 313 |
| 18xSwordmasters of Hoeth | Full command Banner of Sorcery | | 350 |
| 18xSwordmasters of Hoeth | Full command | | 300 |
| Repeater Bolt Thrower | | | 100 |
| Repeater Bolt Thrower | | | 100 |
| Great Eagle | | | 50 |
| Great Eagle | | | 50 |
| Total cost: | | | 2500 |

Table C.5: Case army A4

| Army ID: A4 - High Elves | | | |
|---------------------------------|--|---------------------|-------------|
| Unit name | Equipment | Utility unit | Cost |
| Archmage | Book of Hoeth | | 360 |
| Prince | Armour of Caledor Great weapon Bow of the Seafarer | | 247 |
| Noble | BSB Heavy armour Sacred Incense Potion of Toughness | | 172 |
| 10xArchers | | | 110 |
| 10xArchers | | | 110 |
| 27xSpearmen | Full command War banner | | 243 |
| 10xLothorn Seaguard | | | 120 |
| Lion Chariot | | | 140 |
| 18xSwordmasters of Hoeth | Full command Razor Banner Talisman of Loec | | 355 |
| 14xPhoenix Guard | Full command Banner of Sorcery | | 290 |
| 6xDragon Princes | | | 180 |
| 5xEllyrian Reavers | Bow Musician | | 112 |
| Great Eagle | | | 50 |
| Total cost: | | | 2489 |

Table C.6: Case army A5

| Army ID: A5 - High Elves | | | |
|---------------------------------|------------------|---------------------|-------------------------|
| Unit name | Equipment | Utility unit | Cost |
| Archmage | | | 360 |
| Noble | BSB | | 142 |
| Korhil | | | 140 |
| 30xSpearmen | | | 320 |
| 10xArchers | | | 110 |
| 18xSwordmasters of Hoeth | | | 345 |
| 14xPhoenix Guard | | | 290 |
| Great Eagle | | | 50 |
| | | | Total cost: 1757 |

Table C.7: Case army A6

| Army ID: A6 - Dwarfs | | | |
|-----------------------------|--|---------------------|-------------|
| Unit name | Equipment | Utility unit | Cost |
| Dwarf lord | Rune of Preservation Rune of Resistance Rune of Stone Rune of the Furnace Master rune of Spite Great weapon | Shieldbearers | 271 |
| Runesmith | Master rune of balance Rune of Spellbreaking Great weapon | | 149 |
| Thane | BSB Strollaz rune | | 145 |
| Dragon Slayer | | | 50 |
| 25xDwarf warriors | Full command Great weapon | | 275 |
| 25xLongbeards | Full command Shields | | 325 |
| 10xThunderers | Shields | | 150 |
| 20xHammerers | Full command Master rune of Grugni | | 320 |
| 20xIronbreakers | Full command Rune of Determination | | 310 |
| Cannon | Rune of Forging Rune of Burning | | 130 |
| 10xMiners | Standard bearer Musician | | 125 |
| Grudge Thrower | Rune of Accuracy Rune of Penetration | | 130 |
| Organ Gun | | | 120 |
| Total cost: | | | 2500 |

Table C.8: Case army A7

| Army ID: A7 - Dwarfs | | | |
|-----------------------------|---|---------------------|-------------|
| Unit name | Equipment | Utility unit | Cost |
| 10xThunderers | Shields | | 150 |
| 10xQuarrellers | Shields | | 120 |
| 29xDwarf warrior | Full command Shields Great weapon | | 344 |
| 10xMiner | Steamdrill | Prospector | 145 |
| Grudge Thrower | Handgun/brace 2xRune of Penetration Rune of Accuracy | Engineer | 175 |
| Cannon | Handgun/brace Rune of Forging | Engineer | 145 |
| 19xIronbreakers | Full command | | 277 |
| 23xHammerers | Full command | | 306 |
| 2xOrgan Gun | | | 240 |
| Thane | BSB Master rune of Gromil Rune of cleaving Rune if Striking | | 145 |
| Runesmith | Great weapon Master rune of Balance Rune of Spellbreaking | | 149 |
| Master engineer | Brace of pistols | | 80 |
| Dwarf lord | Great weapon Rune of Resistance Rune of Preservation Rune of Stone | Shieldbearers | 221 |
| Total cost: | | | 2497 |

Table C.9: Case army A8

| Army ID: A8 - Dwarfs | | | |
|-----------------------------|--|---------------------|-------------|
| Unit name | Equipment | Utility unit | Cost |
| Dwarf lord | Master rune of Gromril Rune of Warding Rune of Spellbreaking Rune of Guarding Great weapon | Shieldbearers | 271 |
| Runesmith | Master rune of Fear Great weapon | | 149 |
| Thane | BSB Master rune of Defence | | 130 |
| 25xDwarf warriors | | | 275 |
| 10xThunderes | Shields | | 150 |
| 10xQuarrelers | Shields | | 120 |
| 25xHammerers | Full command | | 330 |
| Cannon | Rune of Forging | | 125 |
| Grudge Thrower | | | 80 |
| Organ Gun | | | 120 |
| Total cost: | | | 1750 |